ARCHIVOS, FICHEROS Y BASES DE DATOS INFORMATICOS

EDITORIAL
DISTRESA, S. A.

JESUS MARIA MINGUET
JOSE MAXIMINO GARCIA
MANUEL MUÑOZ GARCIA

JESUS MARIA MINGUET
JOSE MAXIMINO GARCIA
MANUEL MUÑOZ GARCIA

ARCHIVOS, FICHEROS Y BASES DE DATOS INFORMATICOS

EDITORIAL DISTRESA, S. A.

EDICION 1985

© EDITORIAL DISTRESA, S. A.
DERECHOS RESERVADOS PARA TODO EL MUNDO

Este libro no puede ser, sin la autorización previa y por escrito de Editorial Distresa, S. A., registrado, reproducido, almacenado o transmitido, ni en todo ni en parte, por ningún medio y de ninguna forma, bien sea mecánica, por fotocopia, electrónica, fotoquímica, magnética, electroóptica o cualquier otra.

EDITORIAL DISTRESA, S. A.
Avda. Tenor Fleta, 42
Teléfono (976) 38 51 79
50007 ZARAGOZA (ESPAÑA)

IMPRIME:
SOCIEDAD COOPERATIVA LIBRERIA GENERAL
Pedro Cerbuna, 23
50009 ZARAGOZA (ESPAÑA)

DEPOSITO LEGAL: Z-1516-85

ISBN 84-7613-006-6

PRINTED IN SPAIN - IMPRESO EN ESPAÑA

Proseguimos con nuestra tarea: la publicación de unos libros imprescindibles para todas aquellas personas deseosas de estar al día en todo lo relacionado con la Informática.

Este, ahora en sus manos, amigo lector, titulado "Archivos, Ficheros y Bases de Datos Informáticos" es un tema fundamental y de ahí nuestro interés en sacarlo a la luz.

Es un libro que abarca aspectos muy variados de la materia tratada y además, todo ha sido expuesto con la suficiente agilidad y claridad como para que su lectura, a diferencia de otros libros técnicos, no sea difícil para toda clase de individuos.

Para el técnico, el médico, el maestro, el abogado, el científico, el economista, el periodista... y en general para todas aquellas profesiones liberales, es de una gran utilidad, puesto que al conocer la información y datos necesarios con una gran rapidez, a través de su miniordenador o microordenador, le ofrece unas posibilidades enormes de estar al tanto de todas las novedades de su profesión y así poder competir con mayor preparación en su actividad profesional.

Para la Universidad, para los Centros Oficiales, para la Empresa (pequeña, mediana o grande), es también de una importancia capital: poder acceder a una formidable documentación que le evitará la consulta por los medios tradicionales, ganando con ello tiempo y, por consiguiente, trabajo.

Prescindir de estos medios en nuestra época (en otro lugar la hemos llamado la Era de la Informática) es, al mismo tiempo, inútil y muy caro, porque al final no podremos estar en disposición de conseguir los resultados apetecidos, conforme a nuestras metas trazadas.

Las bases de datos, en continuo aumento en todo el mundo, serán un instrumento que transformará nuestro vivir cotidiano, intercomunicando conocimientos universales a gran número de gentes, que así estarán en condiciones de convertir este planeta en que vivimos, en un mundo mejor.

EL EDITOR

AUTORES

Manuel Muñoz García, Licenciado en Ciencias Matemáticas, graduado en Investigación Operativa, miembro de la Sociedad Española de Estadística, Investigación Operativa e Informática.

Desempeñó el puesto de analista funcional en empresas de ingeniería.

Jefe del Servicio de Mecanización de EPTISA, donde se responsabilizó del diseño de Sistemas de Gestión basados en la utilización de grandes bases de datos. En la actualidad desempeña el puesto de jefe de proyectos informáticos de gestión en Empresarios Agrupados, S.A.

Ha escrito diversos libros y artículos sobre la Teoría de la Informática Básica y su aplicación al mundo empresarial, especializándose en temas relacionados con los problemas existentes para el almacenamiento y gestión de grandes volúmenes de datos.

D. José Maximino García González, Ingeniero de Telecomunicaciones, Profesor Numerario de Tecnología Electrónica en F.P. y Especialista en divulgación de temas informáticos.

Ha trabajado en preparación de textos y enciclopedias para personal no experto en informática.

Traductor de diversas publicaciones técnicas para empresas informáticas (Hewlett Packard, NCR).

Jesús M.ª Minguet Melián es Dr. Ingeniero de Minas y licenciado en Informática. Profesor titular de Lenguajes y Sistemas Informáticos en la Universidad Nacional de Educación a Distancia. Trabaja en Informática desde el año 1963, habiendo ocupado los puestos de Director del Centro de Cálculo de la E.T.S. de Ingenieros de Minas de Madrid; Director Ejecutivo del Centro de Proceso de Datos del Ministerio de Educación y Ciencia, Director Técnico del Centro de Cálculo del INV y Jefe del Departamento de Educación de NCR España. Como docente ha sido profesor de Informática en la E.T.S. de Ingenieros de Minas, Instituto de Informática, CEU de Madrid, Facultad de Informática y E.T.S. de Ingenieros Industriales de la U.N.E.D.

Ha participado en numerosos congresos nacionales e internacionales presentanto ponencias y comunicaciones. Es autor de artículos en revistas, monografías y textos en informática.

PRESENTACION

La mecanización de una empresa u organización, supone el manejo de grandes volúmenes de información relativa a clientes, productos, empleados, sucursales, movimientos bancarios, etc. El ordenador permite un uso exhaustivo de estos grandes volúmenes de datos con una condición básica: los datos deben estar organizados. Y este es el objetivo del libro que tiene Ud. en sus manos.

Los datos se organizaron clásicamente en ficheros o archivos manuales. En la organización informática se ha respetado el nombre de fichero o archivo (File en inglés).

Los ficheros son tratados en la primera parte de esta obra. Así, en el capítulo primero se repasa su evolución a lo largo de la Historia. También se introducen los conceptos básicos de los elementos de información, carácter, registro, indicativo, bloque, etc. Se estudian a continuación los diferentes tipos de ficheros. Se dan consejos prácticos para el diseño de los ficheros. Se termina el capítulo con el estudio de las principales características a tener en cuenta en los ficheros.

El segundo capítulo estudia la relación entre los soportes físicos de los ficheros y las organizaciones de éstos. Se empieza por los medios de acceso secuencial, todos ellos basados en la cinta magnética. En cada caso se estudia cómo se graba la información. Se describe la cinta magnética, el cassette y el cartucho magnético.

A continuación se repasan los medios de acceso directo: el disco, el diskette y el tambor magnético. Se termina con un cuadro resumen de los usos de los diferentes soportes.

En el siguiente capítulo, se introduce la organización de los fiche-

ros distinguiendo entre organización lógica y organización física. Se estudian las organizaciones secuenciales, standard, relativa o directa, encadenada, derivadas del encadenado, regionalizada, particionada, secuencial-indexada e inversa. Se dan al final unos consejos sobre la elección del método de organización.

En el último capítulo de esta primera parte, se estudian los principales métodos de acceso a ficheros: Secuencial, directo, secuencial indexado, particionado y de memoria virtual. En cada método se presentan ejemplos de cómo funcionan. Finalmente se dan unos consejos sobre la elección del método de acceso.

La problemática del manejo de los ficheros se resuelve mediante la técnica de las Bases de Datos, a lo que se dedica la segunda parte de este volumen.

Se inicia el estudio de los conceptos fundamentales sobre bases de datos y sus orígenes históricos, y las causas que dan lugar a buscar nuevas formas de manejo de los ficheros clásicos. Se definen los Sistemas de Base de Datos, el concepto de administrador, la redundancia y las ventajas del uso de las Bases de Datos.

El capítulo sexto trata de la organización de la información. Se comenta la estructura de la información y su relación con los ficheros tradicionales y las bases de datos. También se estudian las relaciones y correspondencia entre datos. Se detallan los conceptos de esquema y subesquemas.

En el siguiente capítulo se describen los modelos de Bases de Datos existentes: jerárquico, en red y relacional, con sus correspondientes ventajas e inconvenientes.

Los tres capítulos siguientes estudian la utilización de los diferentes modelos de Bases de Datos existentes.

Por último se presentan las características de las Principales Bases de Datos existentes en el mercado.

Por todo ello, este libro es fundamental para una adecuada organización de los datos en la empresa antes de iniciar su mecanización.

Jesús M.ª Minguet Melián

PRIMERA PARTE

FICHEROS

CAPITULO I

INTRODUCCION A LOS FICHEROS

1.– ANTECEDENTES

A lo largo de toda su historia, el hombre ha tenido necesidad de usar un medio en el que pudiera almacenar distintas informaciones que posteriormente le pudieran ser útiles, debido a que por su capacidad de memoria no podía retenerlas durante un tiempo indefinido. Por ello se ha servido de diferentes instrumentos de almacenamiento.

Recordemos como en la Edad Media los monjes creaban Bibliotecas, copiando y escribiendo manuscritos sobre los hechos y costumbres de la época. Era la forma de almacenar la información de su tiempo y que es utilizada actualmente por todas las personas interesadas en esa época. En nuestra Era podemos observar cómo el hombre, de acuerdo a sus necesidades, utiliza diferentes medios de almacenamiento. Es muy popular la Agenda, en la que tenemos guardadas las direcciones y números de teléfono, de negocios, de amigos, de familiares, etc., con el fin de que no se nos olviden las entrevistas y visitas que tenemos que celebrar. En las empresas, en las que se necesita tener a mano muchos datos (nombres de clientes, facturas, filiaciones de empleados, etc.), se utilizan los llamados ficheros o archivos, que son carpetas en las que se almacenan todo tipo de datos. Es fácil observar que los ejemplos puestos hasta aquí son manuales, es decir, en todos ellos es el hombre quien organiza sus medios de almacenamiento.

La aparición de la informática ha supuesto una auténtica revolu-

ción en la mayoría de las actividades humanas. La creación de ficheros también se ha visto afectada, de forma muy positiva, con la aparición del ordenador. Va a ser el que se encargue de darnos la información que necesitamos en el momento oportuno. El fichero clásico desaparece y la famosa carpeta con sus fichas, es sustituída por una cinta magnética, un disco, un disquete, etc., que son los nuevos soportes de datos. Tenemos la oportunidad de manejar fácilmente millones de datos con los nuevos medios. En este libro vamos a ir describiendo cómo se construye un fichero, los diferentes tipos que existen de ellos, su organización y la forma de acceder a los mismos, acabando con la exposición de lo que es una Base de Datos.

2.– ELEMENTOS DE INFORMACION

En un ordenador, el elemento más pequeño de información posible es el BIT. El término bit proviene de Binary digit (digito binario) porque sólo puede representar dos valores, el 0 y el 1.

Por consiguiente cuando necesitamos representar variables que puedan tener más de dos valores, es necesario utilizar varios bits. Así con 2 bits se podrían representar $2^2 = 4$ valores; y con n bits podrán representarse 2^n valores diferentes. El número mínimo de bits necesarios para representar cualquier dígito, serán 4.

Sin entrar a fondo en los sistemas de numeración vamos a ilustrar mediante un ejemplo, la correspondencia entre una representación binaria y las representaciones octal y hexadecimal que requieren un menor número de caracteres.

Hexadecimal	1				F				3				(Agrupando bits de 4 en 4)
Binaria	0	0	0	1	1	1	1	1	0	0	1	1	
Octal	0				7				6			3	(Agrupando bits de 3 en 3)

Figura 1.

El Carácter.

Pero el bit es poco útil al usuario corriente, que se mueve en un mundo de dígitos decimales, letras del alfábeto y caracteres especiales (+, −, ×, :, etc.).

Estos caracteres debemos codificarlos en bits para que puedan ser procesados mediante el ordenador.

Si se utilizan 6 bits podremos codificar $2^6 = 64$ caracteres (no se pueden utilizar entonces las letras minúsculas, ni todos los caracteres especiales, ni de control).

El código ASCII utiliza 7 bits, por lo que se pueden representar $2^7 = 128$ carácteres.

Hoy día los códigos más usados y normalizados emplean 8 bits, con lo que se elevan a 256 caracteres el número máximo de elementos que podemos representar.

Este conjunto de 8 bits recibe el nombre de byte u octeto y suele ser la unidad de medida de la capacidad de los dispositivos de almacenamiento.

Campo

También llamado item o dato, es un conjunto de posiciones de memoria que pueden contener la información o valor correspondiente a una variable.

Existen diferentes tipos de campos dependiendo del tipo de variable o dato que van a contener, o del sistema de codificación empleado.

Así existen campos alfabéticos, numéricos, alfanuméricos, de control, de bits, ASCII, de punto flotante, etc.

El elemento de información siguiente es el registro, elemento que por su importancia merece un estudio aparte.

3.– REGISTROS Y FICHEROS

El elemento básico de la moderna forma de almacenar datos es el Registro. En general, el Registro se podría definir como un conjunto de datos heterogéneos, de cualquier longitud, no siendo necesario que exista relación entre ellos.

Antes de seguir, debemos aclarar que la palabra "longitud" se refiere al número de datos o caracteres que puede contener un registro. A veces también se mide la longitud en caracteres. Evidentemente, los datos del registro pueden ser numéricos, alfabéticos, alfanuméricos, etc.

Para el lector, esta definición puede ser "fría", pero unos ejemplos pueden aclararle mucho la idea de registro. Supongamos que una empresa tiene un fichero de empleados. Un registro es el conjunto de los datos referentes a un empleado. En él, se contiene su nombre, apellidos, número de hijos, año de ingreso en la empresa, etc.

Otro ejemplo de registro lo podemos ver en un fichero perteneciente a una revista automovilística. Las características de cada modelo (color, potencia, consumo, etc.) constituyen los registros de dicho fichero. En el fondo, un registro no es más que una secuencia finita de campos, items o datos.

Un campo puede desempeñar diferentes papeles dentro del registro. Así hay campos de *Cantidades* (valor tomado por una variable numérica), *literales* (información no variable para ser reproducida, generalmente títulos), *indicadores o flags* (señal que permite conocer la situación particular del registro dentro de una gama de condiciones posibles), *punteros o pointers* (indicación de la dirección donde se encuentra una información complementaria u otro registro), *códigos* (representación codificada para ahorrar espacio) e *indicativos o claves* de las que hablaremos más adelante.

Aunque en teoría la longitud de un registro puede ser indefinida, en la práctica viene limitada por el sistema o por el soporte. Así muchos sistemas sitúan la longitud máxima en 32.768 caracteres. Otras veces, viene limitada por el soporte (80 caracteres en las fichas perforadas o 128 en muchos disquetes).

Cuando todos los registros de un mismo fichero tienen la misma longitud, se dice que los registros son de *longitud fija*. Se dice que los registros son de *longitud variable* cuando la longitud fluctúa entre unos ciertos límites pero sin sobrepasar nunca una dimensión máxima. Si la longitud del registro es imposible de determinar antes de operar con él, decimos que tenemos un registro de *longitud indefinida*.

Indicativos.

Los datos deben agruparse para formar el Registro, siguiendo un determinado criterio, es decir, tiene que existir un dato en torno al cual se agrupen todos los demás que pertenezcan al mismo registro. A este dato se le conoce con el nombre de *indicativo* y es la clave o base de agrupamiento y de identificación y diferenciación de unos registros con otros. Evidentemente, este indicativo debe ser diferente para cada registro, lo cual obliga, en determinados casos, a una codificación previa.

De nuevo, los ejemplos nos van a aclarar lo expuesto sobre el indicativo. En el caso del fichero de empleados de una empresa, el indicativo podría ser un número de identificación de cada empleado. De la misma manera, cada modelo de automóvil puede tener un número o clave que lo identifique. Los números de identificación de los ejemplos vistos, se codifican previamente para evitar que bien dos em-

pleados, dos modelos de automóvil, etc., tengan el mismo número de identificación.

Registro Lógico y Registro Físico.

La necesidad de estos dos nuevos conceptos se debe a que es el ordenador el que va a manejar los datos de los registros. Los registros Lógicos son los ya definidos anteriormente. A cada uno de los datos que componen el registro lógico se le denomina *campo*, como ya vimos.

Figura 2.

En el diagrama hemos representado un registro lógico que contiene cuatro datos, items o campos. Cada campo o dato estará compuesto de una serie de bits (ceros o unos) de una determinada longitud. Un campo que contenga 8 bits podríamos representarlo así:

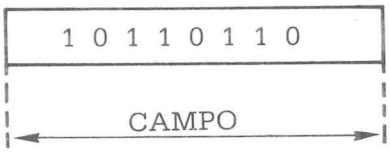


Figura 3.

La información que refleja dicho campo, sigue una determinada codificación para que pueda ser interpretada correctamente por el ordenador.

La aparición del concepto de *"Registro físico"* se debe a que el ordenador, por construcción, para realizar las operaciones de intercambio entre su memoria interna y las memorias externas lo hace siempre con una longitud fija, relacionada con el dispositivo de entrada/salida utilizado. Por lo tanto, esta longitud o espacio puede contenerse en una serie de registros lógicos debidamente agrupados. Esta serie o conjunto de registros lógicos recibe el nombre de *Registro físico* o *bloque*. Se denomina *Factor de Bloque* o *blocaje* al

número de registros lógicos que forman un registro físico. Vamos a representar como ejemplo, un Registro físico cuyo factor de bloqueo sea igual a cuatro, siendo 3 el número de campos de cada registro lógico.

Figura 4.

Como el tamaño del registro lógico no tiene por qué coincidir con el tamaño del registro físico, pueden darse las tres situaciones siguientes:

Registro lógico < Registro físico. Hay que usar *registros bloqueados* de forma que el factor de bloqueo desperdicie el menor espacio posible.

Registro lógico = Registro físico. El *registro* es *unitario*; por lo tanto, el factor de bloqueo es 1, aunque a veces se dice que son registros sin bloquear.

Registro lógico > Registro físico. Se ocupan varios registros físicos por el bloque, de forma que se desperdicie el mínimo de espacio. Se suelen llamar *registros expandidos.*

Ficheros.

Después de todo lo expuesto, estamos preparados para definir lo que es un fichero. A "un conjunto de Registros lógicos perfectamente relacionados entre sí" se le denomina *fichero o archivo.* Es evidente que, siguiendo con los ejemplos expuestos sobre registros, el fichero de empleados de una empresa estará formado por el conjunto de registros lógicos que contienen la información referente a cada uno de ellos.

Determinación de los bloques.

Los registros lógicos que constituyen el fichero se van a agrupar formando bloques. La determinación del factor de Bloqueo o número

de los registros lógicos que van a formar el bloque, va a depender de diversos factores. Al construir el fichero, hay que conocer el soporte o dispositivo en el que se va a almacenar (cinta magnética, disco magnético, disquete, tambor, etc.), la velocidad con que se va a realizar la búsqueda de los datos, la velocidad con que se van a leer o a escribir los mismos, el espacio que se va a ocupar en la memoria, etc.

Para aclarar ésto, pensemos sólo en lo diferente que es disponer para oir música, de un tocadiscos o de un cassette. La selección de una determinada melodía parece más accesible si utilizamos un disco que si utilizamos un cassette o cinta magnetofónica. Con ésta última hay que rebobinar en un sentido o en otro para llegar a ajustar el principio de la melodía que queremos escuchar, mientras que con un tocadiscos sólo hay que colocar la cabeza que lleva la aguja de reproducción, en el lugar de inicio de la grabación de la melodía.

Todos estos factores hacen que se busque el factor de blocaje ideal para optimizar:

a) La ocupación de información en el soporte de la memoria externa.

b) El tiempo de proceso.

Vamos a suponer que disponemos de un fichero grabado sobre cinta magnética. Es totalmente necesario que entre bloque y bloque exista un espacio sin información, para permitir la aceleración y desaceleración que tiene lugar entre lecturas físicas. La longitud de este *espacio interbloques* es fija, por lo que la cantidad de informa-

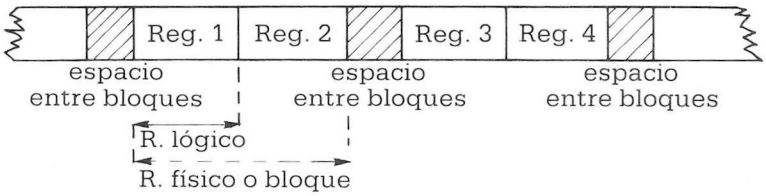


Figura 5.

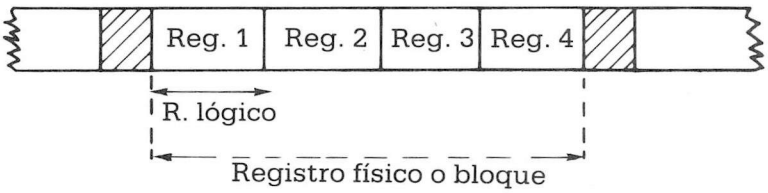


Figura 6.

ción válida existente sobre una cinta, es decir, el número de caracteres leídos por unidad de tiempo, depende directamente del factor de blocaje. Es muy distinto que un bloque contenga, por ejemplo, dos registros lógicos por bloque que cuatro. En este último caso, el número de operaciones de lectura es menor, disminuyendo el tiempo de la operación.

En el caso que sea posible, se usarán los registros no bloqueados, que recordamos son los Registros cuyo factor de Blocaje es igual a 1, es decir, en los que el tamaño del bloque o registro físico coincide con el tamaño del registro lógico.

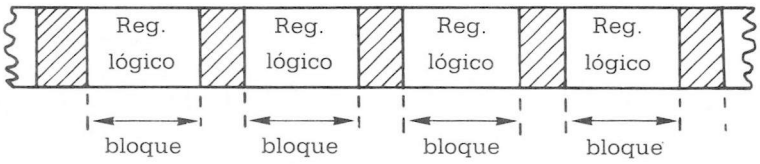


Figura 7.

Hay también ocasiones en que el registro lógico es de un tamaño superior al del registro físico y tiene, por tanto, que distribuirse entre varios de éstos para su almacenamiento, constituyendo lo que hemos llamado *"Registros Expandidos"*. Estos casos aparecen cuando hay una razón, generalmente física, que limita el tamaño máximo del Bloque.

Vamos a exponer dos ejemplos de Registros Expandidos, que aclaren lo dicho sobre ellos. Supongamos que disponemos de fichas perforadas de 80 columnas, en las que se tienen que almacenar registros lógicos de 120 caracteres. Para solucionar este problema, al disponer de fichas de 80 caracteres, hay que meter en dos de ellas los 120 caracteres. Otro caso típico puede ser cuando el tamaño del registro a almacenar sea superior, por ejemplo, al de la pista del disco magnético en el que se va a guardar la información. En este caso habrá que utilizar una segunda pista.

Para que en la memoria interna del ordenador se pueda reconstruir el Registro completo, éste se divide en varias partes llamadas segmentos. Cada uno de éstos, llevará como cabecera unos bits que van a servir como identificadores de cada uno de los segmentos y así poder reconstruir el Registro completo. Como ejemplo, supongamos un registro B dividido en tres segmentos. Cada uno de éstos va a llevar varios campos. Los primeros contienen el código de identificación, la longitud del segmento y el código de control.

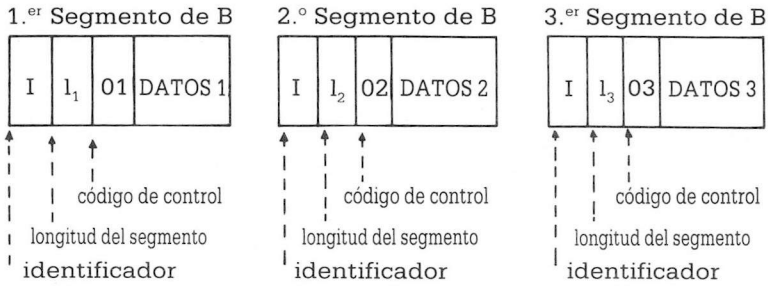


Figura 8.

Con el fin de aprovechar al máximo el espacio, puede utilizarse un factor de bloqueo inferior a la unidad. Así, si los registros físicos admiten como máximo 80 caracteres y los registros lógicos son de 120, pueden cogerse tres registros físicos para almacenar dos registros lógicos. Por lo tanto, el factor de blocaje será de 2/3.

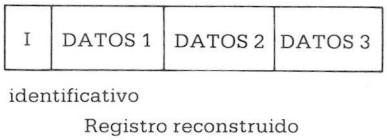


Figura 9.

4.– TIPOS DE FICHEROS

Existen dos clasificaciones de ficheros atendiendo a los criterios siguientes:

a) Según el tipo de información contenida en los ficheros.
b) De acuerdo al movimiento de datos entre la memoria interna del ordenador y el fichero.

Siguiendo el primer criterio, tipo de información contenida en el fichero, se pueden clasificar en:

– Ficheros básicos o de movimientos.
– Ficheros intermedios o temporales.
– Ficheros históricos o maestros.

Los *ficheros básicos* o *de movimientos*, son los que contienen la información de entrada o salida al proceso mediante ordenador, de una determinada aplicación.

Siguiendo nuestro criterio de exponer un ejemplo que clarifique la explicación dada, pensemos en un almacén y en un fichero que contenga los datos de entrada y salida, en un día, de los artículos almacenados en él y que sirve de base al proceso de actualización del control de stock. Este fichero es de movimientos, se utilizará un día y a continuación se destruirá.

Los *ficheros temporales* también llamados *de trabajo* o *scratch*, son aquellos que se crean durante el tratamiento por ordenador de una determinada aplicación. Se utilizan en el proceso de la misma, sirviendo de enlace entre los ficheros básicos y los históricos y destruyéndose dentro del mismo proceso.

Un ejemplo típico de este tipo de fichero, sería el dedicado a acumular las ventas diarias por artículo de los diferentes establecimientos de una cadena de grandes almacenes. Este fichero se obtendrá a partir de los ficheros de movimientos procedentes de cada uno de los establecimientos, y se utilizaría posteriormente para actualizar, en una sola pasada, el fichero maestro de ventas.

Finalmente, los *ficheros históricos* o *maestros*, son aquellos que tienen una vida superior al proceso de una aplicación y contienen los datos finales de dicho proceso.

Un ejemplo típico puede ser el fichero que contiene los datos de la calificaciones finales de los alumnos de un colegio. Este fichero se actualizará anualmente con las calificaciones de los alumnos matriculados, pero habrá que conservarlo permanentemente a fin de poder facilitar certificados de estudios, en cualquier momento, a todos los alumnos que hayan pasado por ese colegio.

Estos ficheros maestros, también denominados *de base* o *permanentes*, al ser de alta estabilidad y duración, deben protegerse fuertemente con el fin de evitar accidentes. Es aconsejable disponer de copias de seguridad actualizadas periódicamente.

Existen tres subclases de ficheros maestros:

Ficheros históricos propiamente dichos: La información no es de actualidad pero refleja la evolución de la información. Son de crecimiento elevado y de poca consulta.

Ficheros de situación: Contienen el estado actual de información que cambia rápidamente (cuentas bancarias, pedidos, inventarios, etc.). Tanto la consulta como la actualización es elevada.

Ficheros de indicativos: Sólo tienen datos indicativos de situación o históricos (Tesauro bibliográfico, productor, departamento, clientes, etc.). Consulta alta y renovación pequeña.

Presentemos un ejemplo concreto que abarque los tres tipos ya definidos. Una red de distribución de accesorios de automóviles informatizada, contiene estos tres tipos de ficheros. El de movimientos contendría la venta diaria de accesorios en cada uno de los establecimientos. Esta venta diaria se recogería en un fichero temporal que se mantendría hasta que con ayuda de él se actualizase el fichero maestro de ventas, que nos daría la información de las ventas totales de la empresa, desde sus inicios.

De acuerdo con el segundo criterio, movimiento de los datos entre la memoria interna del ordenador y el fichero, tenemos la siguiente clasificación:

- Ficheros de procedencia o de entrada.
- Ficheros de destino o de salida.
- Ficheros de procedencia-destino o de Entrada/Salida.

Los *ficheros de Entrada* son aquellos desde los cuales se envían los datos a la memoria del ordenador. Es un archivo que ya se tiene, es decir, ya se había creado y cuya misión es suministrar información a un programa. Como ejemplo de este tipo, pensemos en un fichero que contenga las ventas diarias de un establecimiento que se va a utilizar para actualizar el fichero de ventas anuales del mismo.

Los *ficheros de Salida* son aquellos en los que únicamente se pueden meter datos. El ejemplo es inmediato: un fichero de salida típico es el que contiene las ventas anuales del establecimiento visto en el ejemplo anterior.

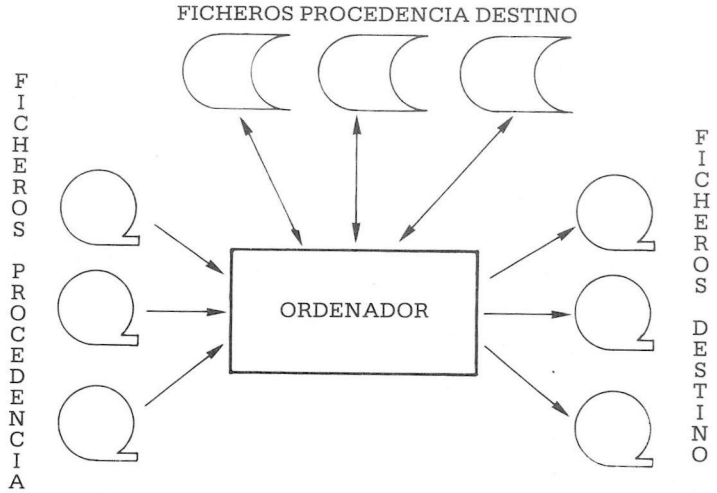


Figura 10. Esquema de ordenador.

Los *ficheros de Entrada/Salida* son aquellos en los que se pueden introducir o sacar datos. Este tipo de ficheros engloba a los dos anteriores, pero es interesante hacer notar que existe una destrucción de datos, por lo que hay que realizar copias del fichero, cuando nos interese la información del mismo, antes de que se reemplace por otra información. Como ejemplo típico de este tipo de ficheros, tenemos el sistema de reservas de billetes de tren. Este sistema contiene ficheros de Entrada/Salida. Se introducen datos al ir a comprar un billete, estos datos actualizan el número de plazas que quedan y a la vez, podemos sacar datos al pedir información sobre las plazas que quedan libres para un determinado trayecto. Otro ejemplo muy típico de este tipo de ficheros son los usados por terminales situados en bancos y cajas de ahorros. Con ellos se pueden realizar operaciones de entrada y salida de datos.

5.– DISEÑO DE FICHEROS

Ya hemos dicho en apartados anteriores, que un fichero estaba constituido por una serie de registros, pero no hemos entrado muy a fondo en su composición. Cuando nosotros queramos construir un fichero hemos de tener en cuenta varios factores. Uno de estos es la forma en que vamos a componer los registros que van a constituir el fichero. Otro factor es el tipo de organización del fichero. En este apartado vamos a estudiar únicamente el primer factor, dejando el segundo para capítulos posteriores.

A la hora de componer los registros que van a constituir nuestro fichero, hay que considerar lo siguiente:

– Longitud de los registros lógicos.
– Número de registros que el ordenador va a leer en una operación.
– Complejidad del tratamiento.

La longitud indica el número de bits o caracteres que forman el registro lógico. ¿Es necesario que todos los ficheros tengan registros lógicos con la misma longitud?. Esta pregunta se puede responder diciendo que depende del tipo de almacenamiento que va a albergar el fichero. Van a existir aplicaciones que utilizan registros lógicos de longitud fija, otras necesitarán registros lógicos de longitud variable e incluso registros lógicos de longitud indefinida.

Los registros de longitud fija son aquellos cuyo tamaño es constante a lo largo de todo el fichero. Se van a utilizar en los casos en que el número de datos contenidos en el fichero no va a sufrir modificaciones en el curso del proceso de una aplicación. Veamos el ejemplo siguiente: Una empresa posee un fichero que contiene los datos re-

ferentes a sus empleados. ¿Qué longitud tienen los registros lógicos que lo constituyen?. Evidentemente la longitud del registro lógico será fija para todos los que componen el fichero, ya que al cambiar alguno de los datos (un empleado se casa, cambia el número de seguridad social, etc.), no va a implicar ningún cambio en la longitud del registro, sólo cambia el contenido de un campo del fichero. Otro ejemplo típico de registro lógico de longitud fija es el fichero que contiene las notas de los alumnos de un curso de un colegio. Todos los alumnos, naturalmente, tienen el mismo número de asignaturas. Con este tipo de registros, los bloques son igualmente de longitud fija para todos los que componen el fichero, tanto para el caso en que los registros no están bloqueados (factor de blocaje = 1) como para los que si lo estén (factor de blocaje diferente a 1). Podemos tener un fichero cuya longitud de registros sea de 20 caracteres y el factor de blocaje sea igual a 3, lo que significa que el ordenador al hacer una entrada lee tres registros lógicos, es decir, 60 caracteres, y en las siguientes operaciones va a leer exactamente lo mismo.

Si utilizamos registros de longitud variable, podemos formar un fichero en el que tanto el tamaño de registro lógico como el del bloque sean variables. No olvidemos decir que el tamaño del bloque tiene un tope máximo a fin de poder reservar en la memoria la capacidad de almacenamiento necesaria para poder realizar una operación de Entrada/Salida. Para que se pueda trabajar adecuadamente con este tipo de registros, la longitud de cada uno de ellos viene indicada por un campo al comienzo del mismo y la longitud de cada bloque también viene dada por un campo de bits.

En las figuras siguientes exponemos cómo son este tipo de registros variables para el caso de que el factor del blocaje sea igual a 1 (no bloqueado) y para registros bloqueados.

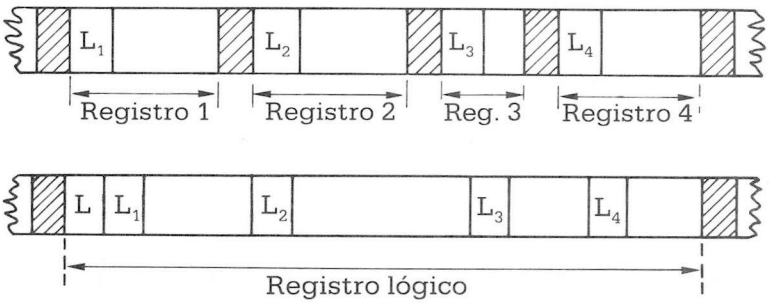


Figura 11.

El diseño de este tipo de registros tiene como objetivo aprovechar al máximo la memoria del ordenador. Al no ser necesario almacenar ni transmitir los campos de datos que no se utilicen y aunque los bits indicadores también ocupan memoria, se alcanza un mejor rendimiento con este tipo de registros que con los de longitud fija, ya que podría haber algunos de éstos que tendríamos que rellenar con ceros para conseguir mantener constante la longitud de todos los registros.

Si tuviésemos que realizar un fichero que contenga las facturas por cliente de una empresa, recurriríamos a este tipo de registro de longitud variable ya que el número de facturas puede cambiar para cada cliente. En ficheros dedicados a labores contables donde los datos son muy variables, también se utilizan este tipo de registros. Sin embargo, hemos de resaltar que son de difícil manejo.

Se pueden emplear también registros de longitud indefinida en la construcción de ficheros. Existen sistemas operativos que permiten la utilización de este tipo de registros y es el programador el responsable de realizar todo lo necesario para que se lleve a cabo un buen tratamiento de los datos.

La localización del fin del registro ha de hacerse analizando los caracteres uno a uno hasta encontrar el carácter que marca el final del registro.

6.– CARACTERISTICAS DE LOS FICHEROS

Para decidir qué tipo de organización debemos dar a un fichero, es necesario conocer algunas de las características inherentes al mismo.

Volatilidad o tasa de renovación.

Viene expresada por la siguiente fórmula:

$$V\% = \frac{\text{N}^{\circ}\text{ de adiciones} + \text{N}^{\circ}\text{ de supresiones}}{\text{N}^{\circ}\text{ inicial de registros}} \times 100$$

Indica el número relativo de registros que aparecen o desaparecen en un tiempo determinado (mes o año). Si es bajo se dice que el fichero es "estático" y si es alto "volátil".

Tamaño.

Es el número de caracteres o bytes que constituyen el fichero. A veces también se mide en registros, sobre todo en los de longitud fija.

Crecimiento.

Determinado por:

$$C\% = \frac{\text{n° de adiciones} - \text{n° de supresiones}}{\text{n° inicial de registros}} \times 100$$

Se dice que un fichero es "estable" si C es próximo a 0. El crecimiento puede ser negativo.

Actividad.

Determina el % de registros que se usan en el proceso normal de una aplicación y viene dada por:

$$A\% = \frac{\text{N° consultas} + \text{N° modificaciones} + \text{N° adiciones} + \text{N° supresiones}}{\text{N° inicial de registros}} \times 100$$

Como vemos, se tienen en cuenta las consultas, modificaciones, adiciones y supresiones, y es uno de los factores más decisivos en el diseño de ficheros.

En el caso de actividad alta, ésta debe estudiarse en partes, es decir, los índices de consulta, modificaciones, altas y bajas, a fin de elegir la organización más adecuada a cada caso.

CUESTIONARIO CAPITULO 1

INTRODUCCION A LOS FICHEROS

1.– ¿Cuál es el elemento más pequeño de información?
2.– ¿Cuál es el código más usado?
3.– ¿Qué se entiende por registro?
4.– ¿Por qué es importante el indicativo?
5.– ¿Qué diferencia un registro físico de un registro lógico?
6.– ¿Qué se entiende por factor de bloqueo?
7.– ¿Qué es un registro expandido?
8.– ¿Qué es un fichero de movimientos?
9.– ¿Cuándo se deben utilizar registros de longitud variable?
10.– ¿A qué se llama volatilidad de un fichero?
11.– ¿Cuándo un fichero es estable?
12.– ¿Qué se entiende por actividad de un fichero?

CAPITULO II

SOPORTES DE FICHEROS

1.– INTRODUCCION

Durante la época en que la construcción de ficheros se realizaba de forma manual, era el papel el medio más utilizado para soportar la información. La llegada del ordenador y el desarrollo de sus posibilidades como medio de tratamiento de ficheros, ha dado lugar a la aparición de nuevos soportes de almacenamiento más acordes con la nueva técnica.

Dejando aparte elementos ya casi abandonados, como los ficheros en tarjetas perforadas, cintas de papel, etc., nos concentraremos en los medios más utilizados en la actualidad: los soportes magnéticos.

Los medios magnéticos de almacenamiento han desplazado a los clásicos por múltiples razones, entre las que destacan:

- Ser más duraderos que los soportes de papel.
- Al poder ser borrados, pueden utilizarse muchas veces.
- Almacenan más datos en menos espacio. Por ejemplo un carrete de 10'5 pulgadas de cinta magnética, contiene unos veinte millones de caracteres, lo que equivale a doscientas cincuenta mil tarjetas perforadas.
- Permiten diferentes estructuras de archivos.
- Son más baratos por unidad de datos almacenados.
- Resultan de más fácil manejo.

2.– MEDIOS DE ACCESO SECUENCIAL

Son los primeros elementos de soporte magnético utilizados. Se denominan de acceso secuencial porque los registros deben grabarse unos a continuación de otros. Todos ellos están basados en la cinta magnética.

Cinta Magnética.

Construida con algún material no magnético, por ejemplo el Mylar, sobre el que se deposita una delgada capa de óxido ferroso que se puede magnetizar. Un punto magnetizado en la cinta representa un valor binario o bit 1; si el punto no esta magnetizado representa un bit 0.

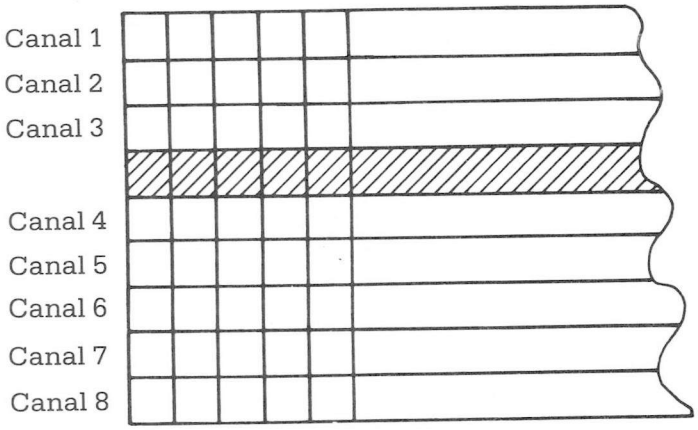


Figura 12.

Para que la grabación de información sobre cinta magnética sea correctamente interpretada por el ordenador, ésta se divide en una serie de canales horizontales que recorren la cinta longitudinalmente.

Al magnetizar los puntos de estos canales tendremos un uno, y si no magnetizamos los puntos tendremos un cero, de manera que la información recogida a lo ancho de la cinta (conjunto de ceros y unos) forma un carácter.

Normalmente, la cinta magnética emplea 7 o 9 canales para grabar dependiendo del tipo de ordenador y de unidad de cinta de que se disponga. Hoy se usan casi exclusivamente las cintas de nueve canales. En la figura 13, el carácter A tiene la información siguiente: 01010001, y el B: 10110010. ¿Qué código siguen los datos almacenados en la cinta?. Pues siguen cualquier código binario que sea utili-

zado por un ordenador. Si almacenamos en la cinta los datos contenidos en la memoria del ordenador, ésta los grabará siguiendo la misma configuración binaria que está en la memoria.

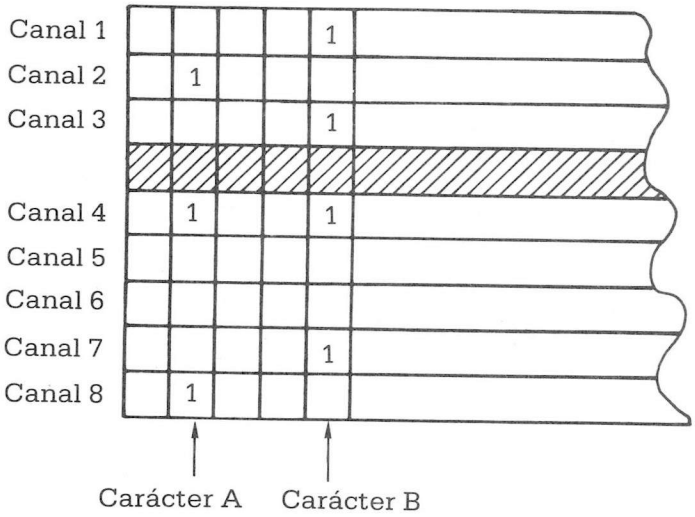


Figura 13.

Método de Paridad.

Para saber si los procedimientos de lectura y grabación son correctos, las unidades de cinta magnética realizan, de forma automática, una serie de verificaciones. Una de las técnicas utilizadas para dicha verificación es la de *Paridad*. Uno de los canales longitudinales de la cinta no contendrá bit de dato, sino que se encargará de alojar los llamados bits de paridad. Normalmente, este canal se sitúa en el centro de la cinta, ya que en esta región habrá menos probabilid de distorsión debido a los estiramientos o deslizamientos, y se podrá leer el bit de paridad con toda exactitud.

La paridad utiliza un bit, llamado *bit de paridad*, que se graba sobre el canal de paridad. La inclusión de dicho bit, o su no inclusión, en cada carácter de la cinta, hace que el número de bits sea siempre par o impar. Veamos un ejemplo de paridad impar. Supongamos que tenemos una cinta de siete canales, tal como la de la figura 14.

Si utilizamos el criterio de paridad impar, como los canales de los caracteres 1 y 3 contienen un número par de bits (4 y 2 respectivamente), hay que añadirles un bit en el canal de paridad para que la

suma total de bits sea impar. El carácter 2, como posee 5 bits de datos, no necesita bit de paridad. A este tipo de paridad se le llama *paridad de carácter.*

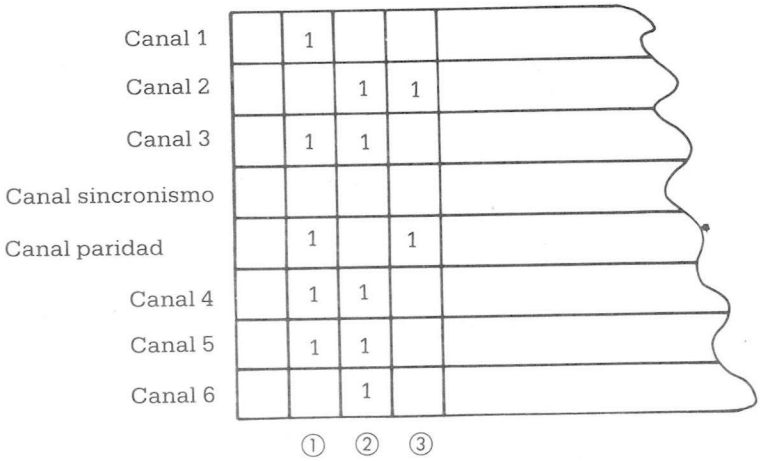


Figura 14.

Existe otro método de paridad, par o impar, llamado *paridad longitudinal,* donde la verificación se realiza en el sentido longitudinal de la cinta en lugar de hacerlo a lo ancho como el de carácter.

Tanto en un método como en otro, la grabación o no de los bits de paridad lo realiza, de forma automática, el hardware de la unidad de cinta. Esta los graba cuando se están escribiendo los datos en la cinta y los verifica cuando se realiza la lectura de los mismos; de esta forma se detectan las pérdidas de bits cuando se realiza la transferencia de datos entre la unidad de cinta y la memoria.

Sincronismo.

Es otra función que depende del hardware de la unidad de cinta y no del programador, al igual que ocurría con la paridad. Volviendo al gráfico anterior, vemos que la cinta tiene en realidad 8 canales: 6 de datos, 1 de paridad y otro de sincronismo. Este se encarga de que el hardware asegure la correcta grabación y posterior lectura de los datos. Cada vez que se graba un carácter, se grava un 1 en el canal de sincronismo. El tiempo que transcurre entre los bits o impulsos de sincronismo es medido por la unidad. La igualdad de dicho tiempo entre los impulsos de sincronismo, determina si los caracteres están

correctamente espaciados en la cinta. Estos impulsos se utilizan también para indicarle a la unidad cuándo debe empezar a aceptar datos.

Densidad de Grabación.

Indica la cantidad de bits que se graban, a lo largo de la cinta, por unidad de longitud. Normalmente se expresa en bits por pulgada, (b p i) o su equivalente caracteres por pulgada (b p i). Según el programa y el tipo de unidad de cintas, la densidad de grabación es variable. Valores típicos son 800, 1600, 3200 y 6.400 b p i.

Velocidad de la cinta.

La unidad de cinta debe moverla a velocidad constante para permitir que la lectura coincida con la grabación de datos efectuada. Dicha velocidad se mide en pulgadas por segundo (i p s).

Existe también la llamada *velocidad de transferencia de datos*, que se obtiene al multiplicar la densidad de grabación por la velocidad de la cinta. Si tenemos una cinta cuya densidad sea de 800 b p i y la velocidad es de 600 i p s, equivale a decir que tiene una velocidad de transferencia de $800 \frac{\text{bit}}{\text{pulg}} \times 600 \frac{\text{pulg}}{\text{seg}} = 480.000$ bits/seg.

Grabación de Ficheros.

Ya hemos visto cómo se graban los bits y los caracteres en una cinta magnética. El paso siguiente es estudiar cómo se graba un fichero. A título de recordatorio, diremos que un fichero se componía de registros lógicos que formaban bloques.

La cinta magnética recoge los registros en forma secuencial, es decir, uno a continuación de otro. Es una grabación en serie, cada registro lleva un campo, dedicado a su identificación, llamado *clave*.

Pensemos ahora qué debe hacer la unidad de cinta para leer los registros de uno en uno. Debe acelerar la cinta desde una velocidad cero (parada) hasta la velocidad de lectura, leer, y cuando acabe de hacerlo, volver a disminuir la velocidad hasta llegar a pararse. Igual ocurre cuando se vayan a grabar los registros de uno en uno. Estos tiempos de aceleración y parada, hacen que la duración de lectura y grabación sea muy larga. Existe, además, otro problema. Hasta que la velocidad de la cinta llegue a su valor de lectura o de grabación, debe pasar, por debajo de las cabezas de lectura o grabación, una cantidad de cinta que no se puede utilizar, luego hay que desaprove-

charla. Estos problemas se solucionan grabando o leyendo los registros, no de uno en uno, sino agrupándolos formando *bloques*, que estarán separados entre si por trozos de cinta sin utilizar llamados *"GAP"*. Observando el siguiente gráfico vemos cómo se organizan bloques de cuatro registros en la cinta.

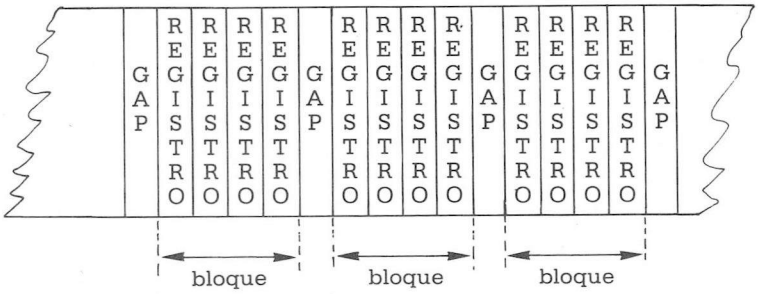


Figura 15.

Para que el operador de la unidad de cinta pueda colocar fácilmente la cinta, se dejan varios metros en blanco al comienzo y al final de cada carrete. A la porción de cinta que se deja al comienzo del carrete se le conoce con el nombre de cabecera y a la del final, cola. Existen, además, dos marcas que son detectadas por la unidad al principio y al final de la cinta, que indican dónde comienza y termina la parte de cinta grabada. La marca de comienzo llamada *BOT (Beging Of Tape)*, está situada a unos 8 metros del inicio de la cinta, y la marca del final, llamada *EOT (End of tape)*, a unos 8 metros del final de ella.

Finalmente, diremos que los carretes poseen una etiqueta exterior a fin de que el personal que trabaja con ellos pueda indentificar el fichero que tiene entre sus manos. Suelen llevar también una etiqueta interior, grabada magnéticamente, con el mismo propósito que la exterior, pero ahora es el programa el que va a indentificar al fichero o ficheros del carrete mediante esta etiqueta.

Fichero de un solo carrete.

Veamos a continuación, la configuración de un fichero que está almacenado en un solo carrete de cinta magnética. Nos ayudaremos del dibujo para detallar mejor esta configuración.

Se inicia, como ya es sabido, por la cabecera y se termina con la

cola. A continuación de la cabecera tenemos dos registros de identificación separados por el GAP. El primero de ellos identifica físicamente al carrete, conteniendo el número que se le asigna a la cinta dentro de la biblioteca de la instalación. Este número no debe cambiarse durante la existencia del carrete.

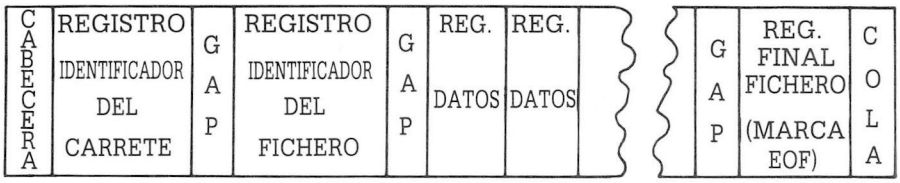


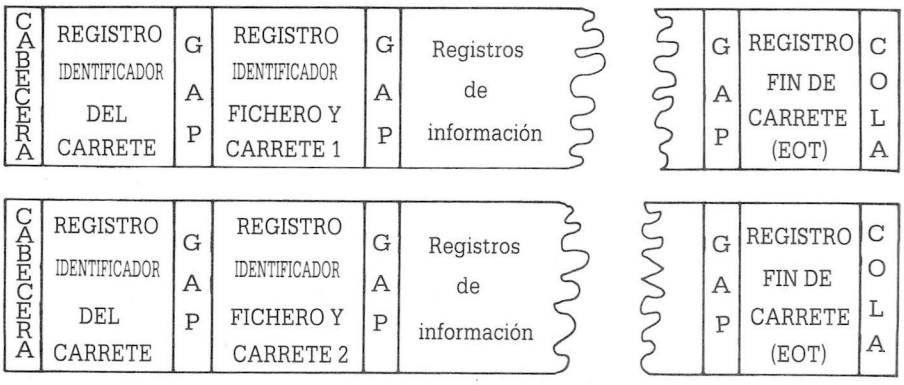
Figura 16.

El segundo registro, o bloque de un registro, contiene el nombre del fichero, fecha de proceso, etc., es decir, la identificación del fichero. De esta manera el programa se asegura el acceso correcto a la cinta deseada.

A continuación tenemos ya los bloques de los registros que contienen los datos, separados por sus GAP. Finalmente tenemos otro registro de fin de fichero o marca de control (EOF = End of tape) al que sigue la cola.

Fichero Multicarrete.

Pueden existir ficheros de tamaño tan grande que necesiten más de un carrete de cinta para su almacenamiento. Pensemos por ejemplo en un fichero que necesite tres carretes de cinta. Estudiemos su configuración:



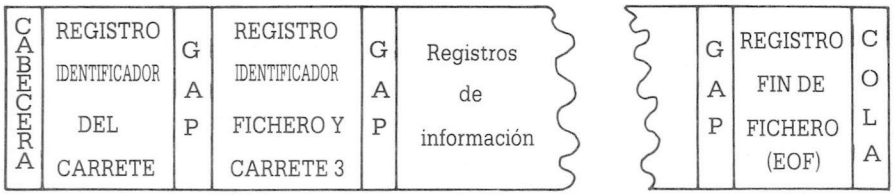


Figura 17.

Este fichero tiene tres *secciones*, una por cada carrete, y el programa que lo procese debe estar preparado para cambiar de forma automática de un carrete a otro. A ésto se le llama enlace de sección a sección. Si no disponemos, en nuestra instalación, de las unidades de cinta necesarias para montar los tres carretes a la vez, el programa debe tener la lógica necesaria para pararse y permitir al operador cambiar los carretes.

En la figura 17, vemos las tres secciones que contienen sus cabeceras, sus colas, los registros que forman los bloques de información y otros tres registros, que son los de identificación. En éstos tres últimos nos encontramos con la identificación del carrete, con la del fichero y su carrete correspondiente y con la de fin de cinta. La marca de control del tercer carrete contiene la marca de fin de Fichero (EOF).

Carrete Multifichero.

En este caso, el tamaño de los ficheros es más pequeño que en el del caso anterior y un solo carrete puede almacenar a varios de ellos. La configuración corresponde a la del ejemplo siguiente:

Figura 18.

Son dos ficheros alojados en un solo carrete con sus correspondientes registros de identificación del carrete, de los ficheros y sus bloques de información y los registros de final de fichero, así como la cabecera y la cola.

A la hora de procesar uno de estos ficheros, hay que buscarlos

desde el principio de la cinta hasta llegar al registro que los identifique. Existen programas que ayudan a esta localización.

Cassette.

En el caso de utilización de ordenadores personales y miniordenadores, el medio de almacenamiento, tanto de programas como de archivos, más utilizado actualmente es el cassette de cinta magnética. Sus ventajas son su bajo coste y su facilidad de manejo y almacenamiento. Está claro que su capacidad de información es mucho menor que la de la cinta magnética, aunque su construcción física es similar.

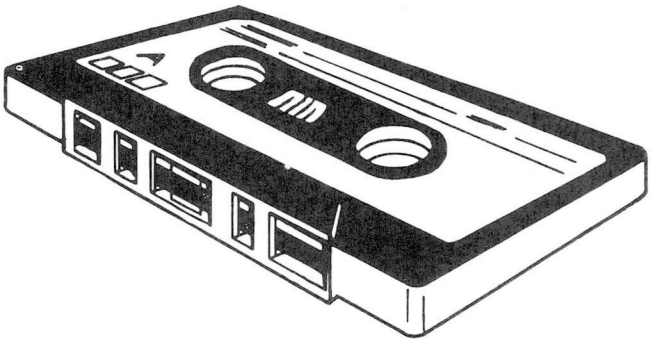


Figura 19. Cinta magnética en cassette.

El ancho de la cinta del cassette es de 4 mm. y su longitud es variable. Existen cintas de 86 y de 15 metros.

Mediante el gráfico podemos aclarar cómo se realiza la lectura y grabación en este dispositivo.

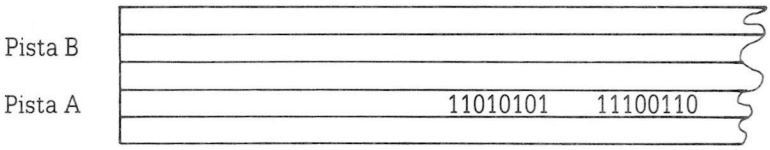


Figura 20.

Existen dos pistas (A y B) para la grabación. Para su reconocimiento, cada una de ellas lleva una etiqueta externa. Si queremos leer o grabar sobre la pista B, por ejemplo, su etiqueta deberá quedar

hacia arriba cuando coloquemos el cassette en su unidad.

Es evidente que el área de grabación es estrecha, por lo que se utiliza el sistema de grabar los bits uno a uno, es decir, en *serie*. En la figura, el área de datos de la pista A contiene información agrupada en caracteres de 8 bits. Podemos dar datos comparativos sobre la densidad de grabación y la velocidad de transferencia en una cinta y en un cassette. En este último la densidad está entre 100 y 800 caracteres por pulgada, mientras que como ya hemos visto, la cinta está entre 200 y 6400 caracteres por pulgada. En cuanto a la velocidad de transferencia, la cinta puede alcanzar cientos de miles de caracteres por segundo y el cassette unos 750 por segundo.

Un cassette, desde el punto de vista de organización de ficheros, sigue los mismos criterios que los dados en el caso de la cinta, es decir, puede haber un fichero en un solo cassette, varios ficheros en un solo cassette o un fichero en varios cassettes.

Cartucho Magnético.

El cartucho de datos contiene una cinta magnética de 4 pistas, cada una de unos 300 pies de longitud. Es necesario llenar una pista de datos, antes de empezar a grabar en la siguiente pista. Como en el cassette, la grabación se efectúa mediante bits en serie.

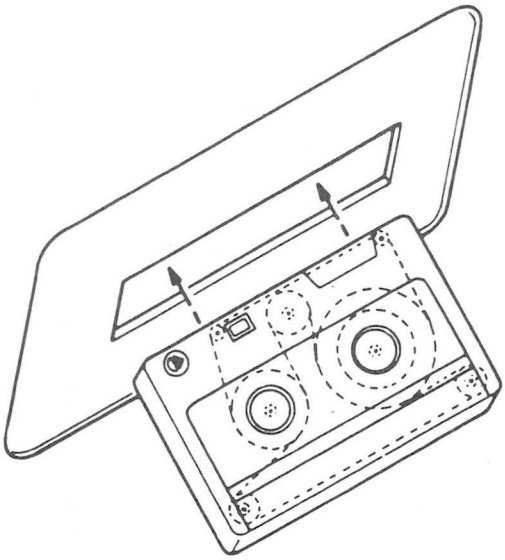


Figura 21. Cartucho magnético.

A diferencia de los otros soportes de cinta magnética, los cartuchos poseen un directorio de archivo, ya que al poder contener varios archivos, es necesario conocer en cuál de las cuatro pistas se encuentra el archivo que se necesita en un momento dado.

3.– MEDIOS DE ACCESO DIRECTO

En muchos casos y sobre todo en un entorno de proceso de datos en tiempo real, es necesario procesar registros uno a uno y sin ninguna relación secuencial con las transacciones inmediatas. Es evidente, por ejemplo, que los clientes que utilizan un cajero automático, no lo hacen en el orden que correspondería a sus números de tarjeta de cuenta. Es preciso buscar un medio de almacenamiento que permita acceder muy rápidamente al registro deseado, sin pasar por todos los anteriores, a fin de reducir el tiempo de las operaciones.

Los dispositivos de acceso directo deben permitir que se pueda acceder a cada registro, en aproximadamente la misma cantidad de tiempo.

Veamos un ejemplo. Consideremos el siguiente archivo de clientes de un banco:

01	JESUS LOPEZ
02	LUIS PEREZ
03	FRANCISCO GARCIA
04	JESUS MINGUET
05	JOSE MAXIMINO
06	JOSE UMBRAL
07	PILAR JIMENEZ
08	OCTAVIO MUÑOZ
09	MARIA CALLE
10	ESTHER GONZALEZ
11	PEDRO PEIRO
12	JUAN PARDO
	ETC.

Si el archivo se encuentra en un soporte secuencial (cinta) y hay que buscar datos de Pedro Peiró, al tener que recorrer los 10 registros anteriores, el tiempo empleado puede ser no aceptable, y muy diferente del usado para atender a Luis Pérez.

Para conseguir un tiempo más aceptable, el archivo se puede dividir en bloques o segmentos. Cada uno de ellos tiene un punto de entrada, es decir, cada bloque tiene su propia dirección identificativa. El archivo podría quedar en la siguiente forma:

1° Punto de entrada	01 JESUS LOPEZ	
	02 LUIS PEREZ	1er bloque
	03 FRANCISCO GARCIA	
2° Punto de entrada	04 JESUS MINGUET	
	05 JOSE MAXIMINO	2° bloque
	06 JOSE UMBRAL	
3er Punto de entrada	07 PILAR JIMENEZ	
	08 OCTAVIO MUÑOZ	3er bloque
	09 MARIA CALLE	
4° Punto de entrada	10 ESTHER GONZALEZ	
	11 PEDRO PEIRO	4° bloque
	12 JUAN PARDO	
5° Punto de entrada	ETC	

De esta forma para obtener el registro de Pedro Peiró se accede directamente al 4° bloque, que se carga en memoria y se analiza secuencialmente hasta localizarlo.

Los medios de acceso directo permiten la segmentación de los archivos en bloques de registros, cada uno de ellos con su propia dirección. El tamaño del bloque está limitado por construcción física y es constante en todo el archivo.

Como además del acceso directo, también permiten el tratamiento secuencial, los medios de acceso directo han copado prácticamente todos los elementos de almacenaje de datos, relegando a las cintas magnéticas, casi exclusivamente, a tarea de respaldo o seguridad de archivos y de carga de programas.

Disco Magnético.

El disco magnético es muy parecido al musical. Se puede grabar o leer sobre las dos caras del mismo, mediante las cabezas de lectura/grabación, situadas en un brazo.

Recibe el nombre de *"pista"* la superficie del plato o disco recorrida por la cabeza durante una revolución del disco.

En estas pistas no sólo se van a grabar los datos sino también las etiquetas de identificación del disco, fichero, etc., como ocurría en la cinta. El acceso directo a los datos se logra posicionando el brazo del soporte de las cabezas en las pistas que se deseen.

Es muy importante indicar que todos los discos van protegidos por unas cubiertas protectoras, sobre las que no se graba y que evitan la "contaminación" de las superficies (polvo, huellas de dedos,

etc). Esta "contaminación" puede conducir a grabaciones y lecturas incorrectas.

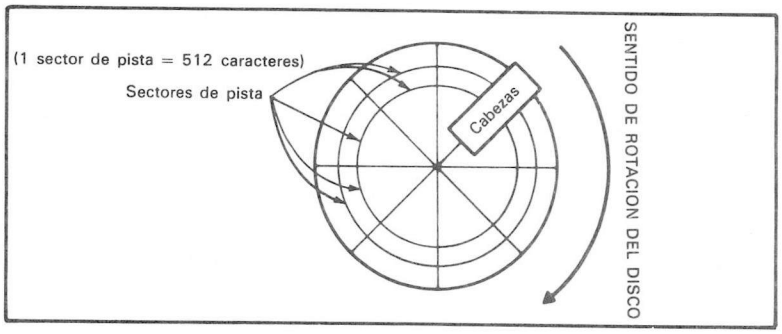


Figura 22. Sectores en un disco.

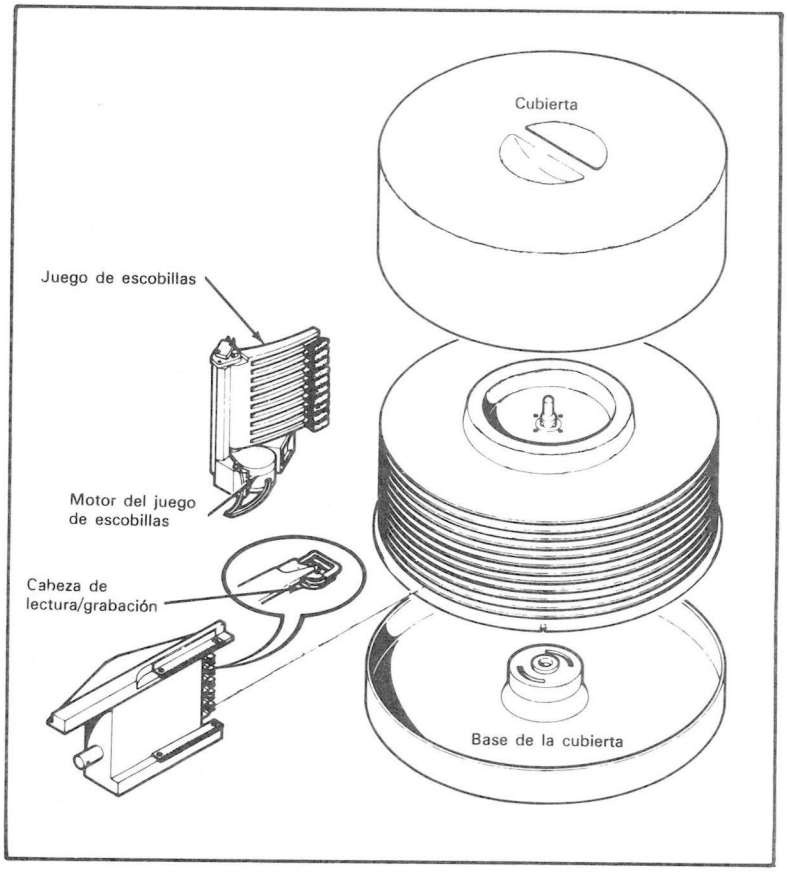


Figura 22-Bis. Pila de discos NCR-957-1.

Ficheros en un Disco Magnético.

Un fichero tiene una estructura, dentro del disco, que se puede descomponer en las siguientes unidades, en orden creciente de tamaño:

- Carácter = Un grupo de bits.
- Sector = Un grupo de caracteres consecutivos.
- Pista = Cada pista alberga a varios sectores.
- Plato o disco = Cada disco contiene un cierto número de pistas.
- Paquete de Discos = Unidad que contiene varios discos con una capa protectora. Si tuvieramos un paquete de discos con cuatro de ellos, es importante destacar que poseerá un brazo con seis cabezas de lectura/grabación, una por cada cara de los discos (recordemos que hay 2 caras de protección).

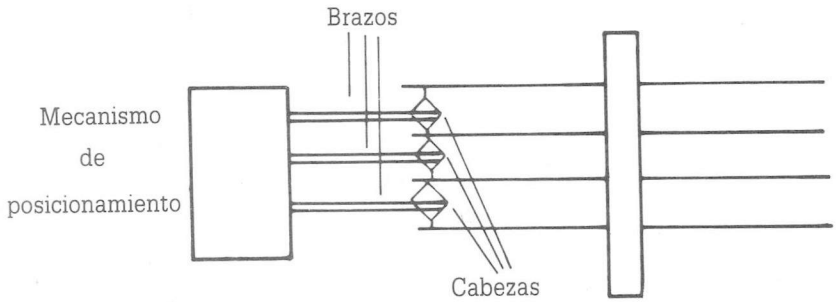


Figura 23.

La moderna tecnología proporciona paquetes de discos con cabezas incorporadas, en un módulo separable de datos hermético a la contaminación.

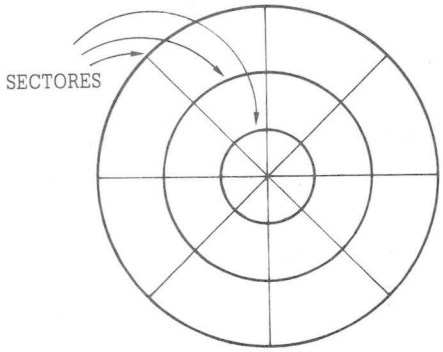


Figura 24.

Cada uno de los sectores de cualquier pista, contiene la misma cantidad de información y aunque las pistas centrales del disco son más cortas que las exteriores, tienen una mayor densidad de grabación.

En este caso tenemos dos pistas con ocho sectores cada una.

El sector va a determinar la longitud de los bloques en el disco. Generalmente un bloque es un sector, pero eso va a depender del tipo de ordenador. Gracias a que estos bloques son leídos y almacenados en la memoria interna del ordenador, de uno en uno, el disco puede ser un medio de procesar ficheros secuenciales y directos.

Se denomina *cilindro* al conjunto de pistas de los diferentes platos que se encuentran a la misma distancia del eje. Por ello todos los datos contenidos en esas pistas son accedidos sin necesidad de mover los brazos que llevan las cabezas de lectura/grabación. Esta ventaja se aprovecha al organizar los ficheros, de forma que un archivo se va llenando por cilindros adyacentes, disminuyendo de esta forma los movimientos del brazo.

Los ficheros en disco también se pueden dividir en *secciones*, que pueden estar compuestas por sectores contiguos del disco, y un fichero puede ocupar dos o más secciones.

En el caso de que un disco contenga varios ficheros o secciones de ellos, es imprescindible crear un *Directorio de Ficheros* del disco, que va a contener la siguiente información:

a) Nombre de los ficheros.
b) Situación de los mismos en el disco.
c) Fecha de grabación.

Conviene resaltar la diferencia que existe entre *Sector* y *Sección*. El sector es una división física de la superficie del disco, mientras que la sección es una división lógica de fichero y su tamaño es variable y asignado por el programador.

Disquete.

También conocido como disco flexible o Floppy-Disk, es un medio de fichero similar al disco magnético. Su tamaño es menor que éste último así como su capacidad de almacenamiento. Se utiliza en miniordenadores y ordenadores personales y con él se pueden procesar ficheros secuenciales y de acceso directo. Lleva una etiqueta exterior para su identificación y una cubierta protectora contra agentes contaminantes.

Aunque existen diversas medidas de disquetes, los más comunes son de ocho pulgadas, y dependiendo de la densidad de graba-

ción, pueden almacenar de 240.000 a 480.000 caracteres.

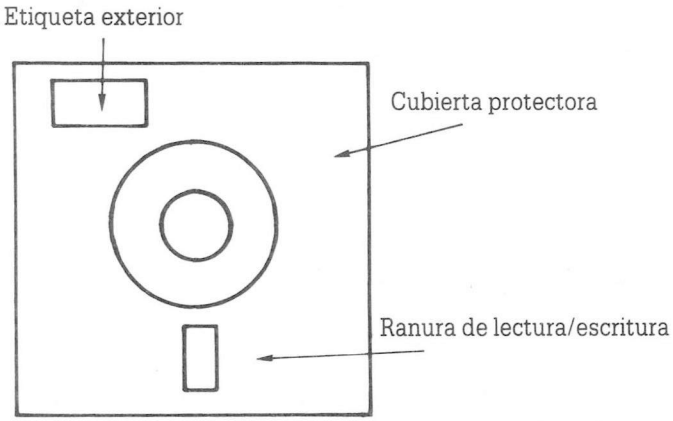


Figura 25.

Tambor Magnético.

Es un medio de fichero que tiene una gran capacidad de almacenamiento, por lo que se empleaba para ficheros de gran tamaño. Se puede utilizar para proceso secuencial y directo. La información se graba o se lee en las pistas de la superficie del tambor. Cada una de estas pistas tiene su propia cabeza de lectura y grabación.

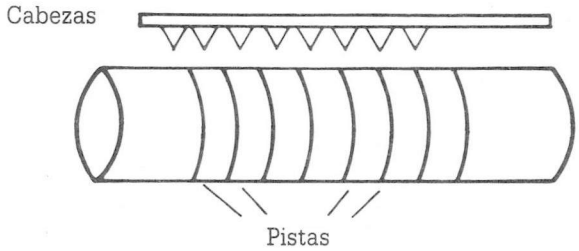


Figura 26.

Para efectuar esta lectura y grabación, el tambor debe estar girando a una velocidad determinada. Esta velocidad depende del tipo de tambor. Tiene como inconveniente frente a la cinta y al disco, que no se puede intercambiar tan fácilmente. La pista se divide en sectores (normalmente dos) y cada sector puede contener un bloque de registros. Los ficheros se pueden dividir en *Segmentos* a los que se accede de forma individual.

Para localizar los ficheros o parte de ellos, el tambor utiliza directorios como el disco.

4.– USO DE LOS SOPORTES

No todos los medios de almacenamiento permiten todos los tipos de archivo. Por ello vamos a presentar en la siguiente tabla las relaciones entre tipo de archivos y periféricos.

	TIPOS DE ARCHIVO		
	ENTRADA	SALIDA	ENTRADA/SALIDA
Papel (Tarjeta, cinta, MICR, OCR)	Entrada Original (proceso en lotes)	Archivos maestros actualizados y de respaldo (proceso en lotes)	No aplicable
Magnéticos secuenciales (cinta, cassette)	Entrada original. Archivos de transacciones. Archivos maestros para actualización (proceso en lotes)	Archivos maestros actualizados y de respaldo (proceso en lotes)	Muy poco práctico
Magnéticos de acceso directo (disco, diskette, tambor)	Archivos maestros. Archivos de transacciones para actualización	Archivos maestros actualizados (proceso en lotes)	Archivos maestros actualizados (proceso en lotes y en tiempo real)
Tarjeta de banda magnética	Entrada de informes	Creación inicial de maestro	Archivos maestros actualizados (proceso en lotes)
Impresora	No aplicable	Informes y listados de respaldo	No aplicable
Terminales	Entrada de transacciones (proceso en tiempo real o en lotes remotos	Respuesta a consultas en tiempo real	No aplicable

CUESTIONARIO CAPITULO 2

SOPORTES DE FICHEROS

1.– ¿Por qué se utilizan principalmente soportes magnéticos?
2.– ¿Cómo se graba la información en una cinta magnética?
3.– ¿Qué se entiende por paridad impar?
4.– ¿Qué es un "gap"?
5.– ¿Qué diferencia un EOT de un BOT?
6.– ¿Cómo se maneja un fichero multicarrete?
7.– ¿Cuál es la diferencia de lectura de un carácter entre la cinta magnética y el cassete?
8.– ¿Por qué son necesarios los medios de acceso directo?
9.– ¿Para qué se utiliza un Directorio de ficheros?
10.– ¿Qué se entiende por cilindro?

CAPITULO III

ORGANIZACION DE FICHEROS

1.– ORGANIZACION LOGICA Y ORGANIZACION FISICA

Ya conocemos la estructura de registros y bloques, así como los medios en los que vamos a almacenar nuestros ficheros. El siguiente paso es conocer la Organización de un fichero, es decir, qué técnicas se van a utilizar para identificar, almacenar y recuperar los datos. Ya hemos hablado de ello en el anterior capítulo, al estudiar los medios de almacenamiento, pero de una forma muy superficial.

Para explicarlo con más detalle, vamos a diferenciar la organización lógica de la organización física.

Organización Lógica.

La organización lógica se refiere a la forma en que el que va a crear los datos, el programador, los ve, independientemente de cómo estén físicamente almacenados en el medio utilizado como memoria exterior (cinta, disco, tambor, etc.). La forma de definir esta organización se basa en el conocimiento que el programador tenga del problema que quiere resolver, independientemente de la forma en que se almacena. El programador se plantea si necesita que sus registros sean tratados de forma secuencial o de forma directa, conocida también como forma al azar o *random*. Desde el punto de vista de la Organización Lógica, existen dos formas de tratar los registros:

1) *Tratamiento Secuencial:* Los registros se localizan en el mismo orden o secuencia en que se almacenaron en el fichero. Por

ejemplo, si se quiere tratar el registro 29, primero hay que leer los 28 anteriores.

2) *Tratamiento Directo:* Se puede localizar un registro sin pasar por los que le preceden, es decir, se puede alcanzar directamente su posición física en medio del almacenamiento. Esto trae consigo una mayor rapidez en la localización de la información que en el tratamiento secuencial.

Organización Física.

La organización física es la forma en que los registros lógicos se colocan dentro del dispositivo de memoria externa. Para definir esta organización, hay que conocer el tipo de dispositivo de almacenamiento, factor de bloqueo, etc. Existe una relación evidente entre la organización física y el registro físico o bloque. ¿Existe también alguna relación o nexo de unión entre la organización lógica y la física?. Debe existir, ya que el programador tiene que tratar los datos de su programa (organización lógica) que están almacenados de alguna manera en un dispositivo (organización física). Esa unión se realiza mediante los métodos de acceso a ficheros que veremos más adelante, y que son unas rutinas que forman parte del sistema operativo del ordenador.

La organización física se compone de una serie de técnicas (llamadas también organizaciones), cuyo objetivo es almacenar y localizar registros físicamente en los medios de memoria externa.

Como en los dispositivos de almacenamiento de acceso directo, se pueden organizar los ficheros de muy diversas formas, el problema es cómo grabar los datos al crear el fichero de forma que cada registro tenga una ubicación física unívoca.

Para elegir la organización más adecuada hay que tener en cuenta tanto las características de actividad, volatilidad, tamaño, etc., del fichero, como los tratamientos que van a utilizar las aplicaciones.

En la figura representamos la diferencia entre organización física y lógica de los datos.

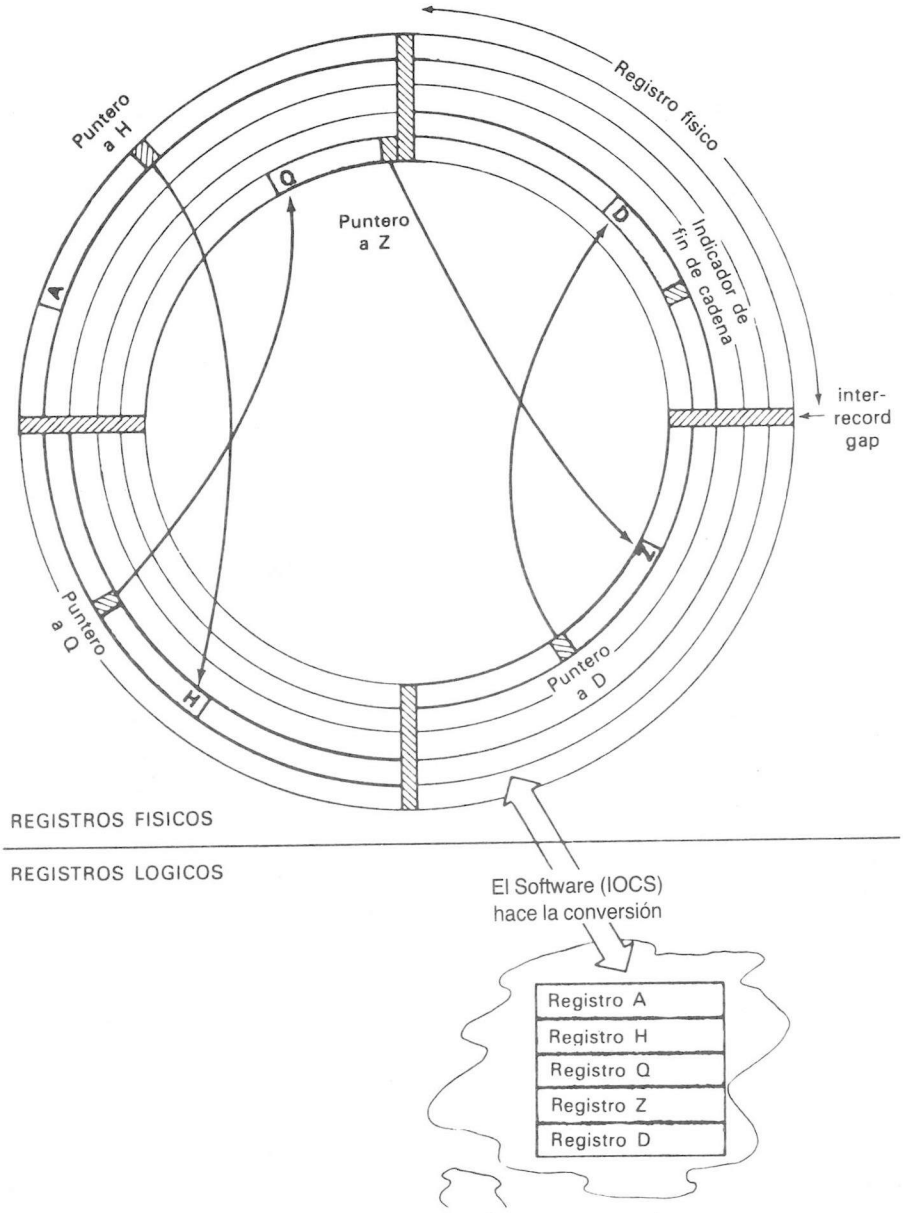


Figura 27.

Veamos a continuación las organizaciones de ficheros más corrientes.

2.– ORGANIZACION SECUENCIAL

Es la más sencilla de todas las organizaciones, en la que los registros se graban físicamente en el dispositivo unos a continuación de otros, generalmente de acuerdo con la secuencia de sus claves de identificación. Si queremos leerlos o actualizarlos hay que hacerlo en el mismo orden en que se grabaron, es decir, su tratamiento lógico tiene que ser asímismo secuencial.

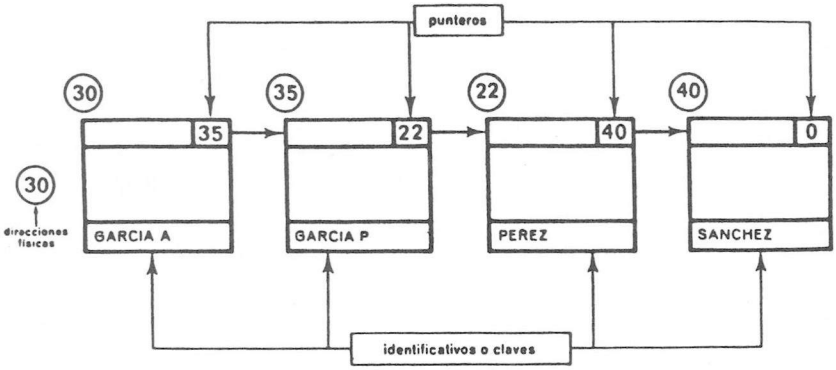


Figura 28.

Ejemplos de ficheros que utilizan esta organización, son los almacenados en cinta magnética y en ficha perforada, aunque también pueden usarse en dispositivos de acceso directo.

Si las claves deben estar ordenadas, hay que hacer que los registros entren en dicho orden al crear el fichero. La inserción de un registro de clave intermedia, obliga a un desplazamiento mediante reescritura, de todos los registros posteriores al que queremos intercalar. La eliminación de un registro obliga a un proceso parecido.

Si se quiere ampliar, hay que dejar espacio al final del archivo. Este espacio se define como extensión del archivo o piggyback. El archivo secuencial no debe tener bloques vacíos intercalados con los de información.

La organización secuencial es buena para un proceso secuencial y puede ser usada para archivos de procedencia o de destino en operaciones en batch. En los casos en que deben tratarse la mayoría de los registros del fichero, es una organización muy económica en soporte y tiempo de proceso.

Si el proceso es al azar, este tipo de organización sólo es válida para ficheros muy pequeños.

3.– ORGANIZACION DE ARCHIVOS STANDARD

De hecho es una variante de la anterior y algunos autores la consideran la misma. La distinción está en que en el secuencial la unidad es el registro, mientras que en la standard es el bloque.

Al utilizar este tipo de organización, los bloques, que contienen los registros lógicos que los forman, se presentan a la hora de procesar el programa en la misma secuencia en la que se grabaron; pero los registros contenidos en los bloques, pueden o no seguir una clasificación, bien numérica o alfabética. Pensemos que estamos realizando un proceso de clasificación, es decir, tenemos un fichero con sus bloques y registros, pero estos últimos no están clasificados por orden alfabético. Con esta organización standard y siguiendo un programa de clasificación, podemos obtener un fichero en el que sus registros ya aparezcan debidamente ordenados. Este tipo de organización puede utilizar como medio de almacenamiento, cinta magnética, disco y tambor.

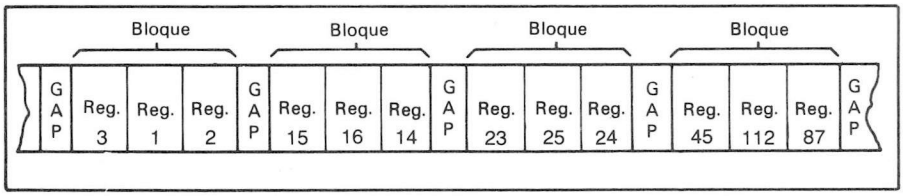


Figura 29. Bloques de registros en una cinta magnética.

Vamos a introducir aquí un concepto que utilizaremos más adelante al hablar de la organización encadenada, pero que ya necesitamos. Se define como *"Area de Rebasamiento"* a una parte o área de almacenamiento magnético, que permite la ampliación de un fichero. Si tenemos un fichero lleno y necesitamos incorporarle nuevos datos, éstos se grabarán en el área de rebosamiento del fichero. Hay que utilizar una serie de técnicas y métodos para conectar el fichero con el área de rebosamiento y poder procesar ésta cuando se necesite. La organización standard no utiliza estos métodos y no puede usar áreas de rebasamiento. El fichero standard se puede ampliar de dos formas:

a) Porque el programador la haya previsto y deje espacio al final del fichero o sección del mismo.

b) Copiando el fichero en un medio de mayor capacidad que el primitivo.

La organización standard ofrece las mismas posibilidades de utilización que las secuenciales.

4.– ORGANIZACION DIRECTA O DE ARCHIVOS RELATIVOS

El programador crea una relación perfectamente definida entre la clave indicativa de cada registro y su localización física dentro del dispositivo de almacenamiento. Los medios más utilizados para el acceso directo son: disco magnético, tambor y disquete.

El registro, que se ha grabado físicamente en un sitio determinado mediante la relación, es localizado con posterioridad de la misma manera. Esta relación se debe mantener siempre aunque se actualicen los registros del fichero.

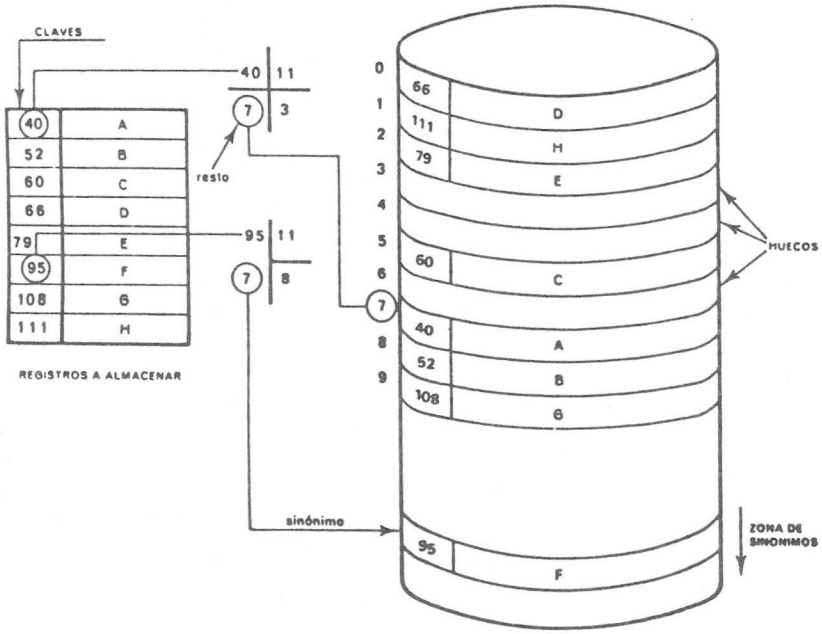


Figura 30.

En cuanto al tratamiento lógico, es también directo, es decir, que los registros se localizan directamente sin tener en cuenta los precedentes.

¿Qué tipos de ficheros buscan este tipo de organización?. Pues aquellos que no pueden permitir la utilización de la organización secuencial o cuando el tiempo para localizar un registro individual-

mente necesita ser muy pequeño, como en las aplicaciones de consulta y bancarias.

En cierta manera, esta organización se basa en la división del espacio disponible para el fichero en celdas que contengan un solo registro, de manera que a cada clave distinta le corresponda una celda.

La relación entre la clave y la dirección, se obtiene mediante el "algoritmo de direccionamiento", que estudiaremos con más detalle en el próximo capítulo.

Es el mejor método de organización cuando el proceso es al azar. El óptimo se da cuando las claves son númericas del 1 al n, con registros de longitud fija y sin huecos.

Los inconvenientes principales son:

- La relación entre clave y dirección física la define el usuario. El sistema operativo no detecta los errores de cálculo de dirección y el usuario debe de comprobar si el registro accedido es el que se busca.
- Cuando el rango de claves posibles es muy superior al real del fichero (por ejemplo el D.N.I. para empleados de una empresa), hay muchos espacios desaprovechados.

Esta organización se utiliza con los archivos maestros de entrada/salida en procesos en lote y en línea.

5.– ORGANIZACION ENCADENADA

La técnica utilizada para organizar un fichero encadenado consiste en conseguir que los bloques del fichero se encadenen entre sí de forma automática. Es el software ejecutivo el encargado de realizar este trabajo, por lo que no es obligatorio que los bloques consecutivos de los registros estén unos junto a otros.

Podemos definir el "encadenamiento" como la técnica de enlazar bloques de registros lógicamente relacionados. Al conjunto de bloques o registros enlazados se le conoce con el nombre de CADENA. Asímismo vamos a definir el PUNTERO como un campo especial, incorporado a la estructura del registro o bloque, cuya función es ser un instrumento de referencia. Mediante él, sabremos qué registro o bloque de registros es el siguiente al que habrá que acceder. El final de una cadena se puede indicar con una marca de fin de cadena. En la figura podemos darnos una idea de la configuración de un fichero encadenado.

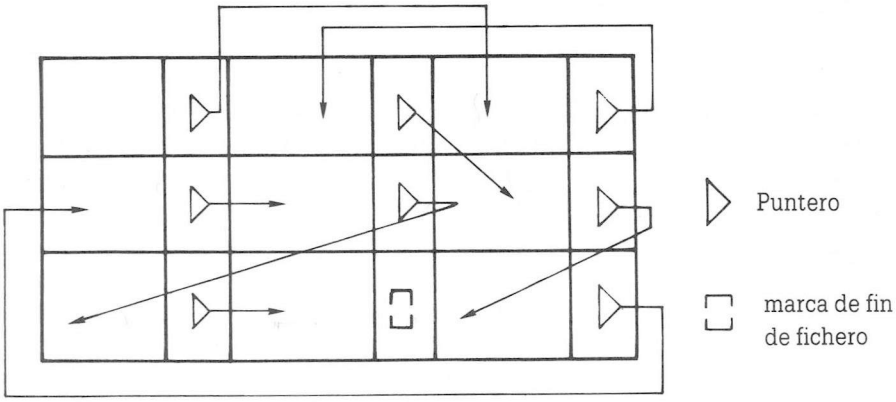


Figura 31.

Los archivos que utilizan esta organización son normalmente de entrada/salida y en tiempo real. De esta forma la organización de alguna manera permite el acceso directo.

Hemos hablado anteriormente de lo que era el Area de Rebosamiento. Si necesitamos tener un fichero con este tipo de área, podemos utilizar la organización encadenada para insertar nuevos registros o suprimir los antiguos. Para conseguir ésto, el fichero queda dividido en dos áreas: una que contiene el fichero principal y otra que contiene el área de rebosamiento. El encadenamiento va a enlazar bloques de registros entre el archivo principal y el área de rebosamiento. En la figura vemos como queda organizado este tipo de ficheros.

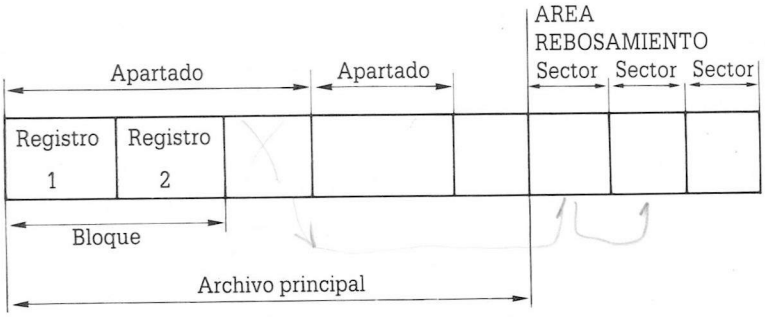


Figura 32.

El "apartado" es una división lógica del fichero cuya misión es ahorrar tiempo en la ejecución de un programa, limitando la búsqueda de registro a un solo apartado, cuando queremos insertar un registro en el fichero, en lugar de tener que buscar en todo él. Los apartados los diseña el programador. Veamos un ejemplo de encadenamiento en este tipo de fichero.

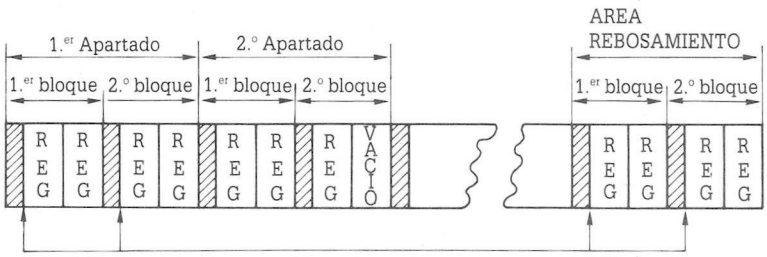


Figura 33.

El 1.° y el 2.° bloque del primer apartado, están encadenados. El 2.° bloque está encadenado con el primer sector del área de rebasamiento y éste con el 2.° sector. Si el 2.° bloque del primer apartado se llena, aparece el rebasamiento y el software localiza el primer sector disponible del área de rebasamiento. Si éste se llena, se pasará al 2.° sector.

En el 2.° apartado, los bloques 1 y 2 están encadenados pero no hay encadenamiento con el área de rebasamiento porque el 2.° registro del 2.° bloque está vacío.

Para terminar este aspecto, vamos a ver cómo se puede insertar un registro en un fichero encadenado. La forma de hacerlo es mediante un desplazamiento (PUSH DOWN) de todos los registros que siguen al que se vaya a insertar. Para realizar este desplazamiento es necesario asignar una serie de espacios vacíos al fichero. Para esto es muy útil emplear los "apartados" y dejar en ellos espacios.

Supongamos que tenemos un apartado como el de la figura.

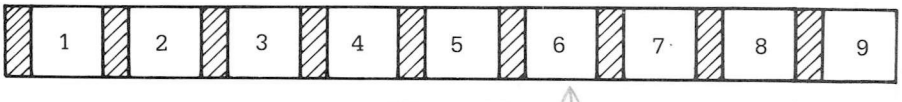


Figura 34.

Nos falta el registro n.° 6. Para insertarlo desplazamos el 7 y el 8 y colocamos el 6. ¿Qué ocurre con el 9?. Si el apartado no ha previsto un espacio de inserta ción, el punto del 9 lo "encadenaría" con un re-

gistro del área de rebasamiento. Es importante observar que habrá que cambiar los punteros de encadenamiento de los bloques afectados por el desplazamiento.

6.– ORGANIZACIONES ESPECIALES DERIVADAS DEL ENCADENADO

Los registros de un archivo pueden ser independientes o estar relacionados unos con otros. En la mayoría de los archivos de uso normal, cada registro del archivo se encuentra solo, está individualizado y no tiene ninguna referencia cruzada entre ellos. Ejemplo típico de ello son los ficheros de personal, cuentas corrientes, etc.

Pero a veces es necesario relacionar unos registros con otros. Así puede interesar, por ejemplo, poder relacionar todas las cuentas de un mismo cliente, o todos los empleados de un mismo departamento. Para ello la estructura del archivo ya no es lineal. Existen unos registros principales, denominados *"maestros"*, de los que pueden depender varios sub-registros, aunque cada sub-registro "sólo puede depender de un maestro". Esta estructura se llama *"en árbol" (tree structure)* y está representada en la figura siguiente.

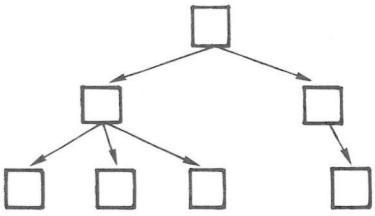


Figura 35.

Si cada registro puede depender o relacionarse con varios registros, tenemos una *"estructura en red"*, como la de la figura.

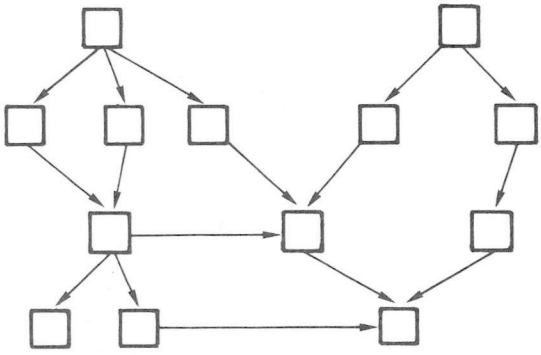


Figura 36.

Estas estructuras de archivos en árbol y en red pueden utilizarse con bases de datos o mediante el método de encadenamiento, ya que los punteros permiten la conservación de las claves lógicas de los siguientes registros. Es evidente que cada registro puede utilizar varios punteros, tantos como registros a relacionarse.

Otra variante de archivo es la *"estructura en anillo"*, en la cual la cadena no termina con la marca fin de cadena, sino con un puntero hacia el primer registro de la cadena.

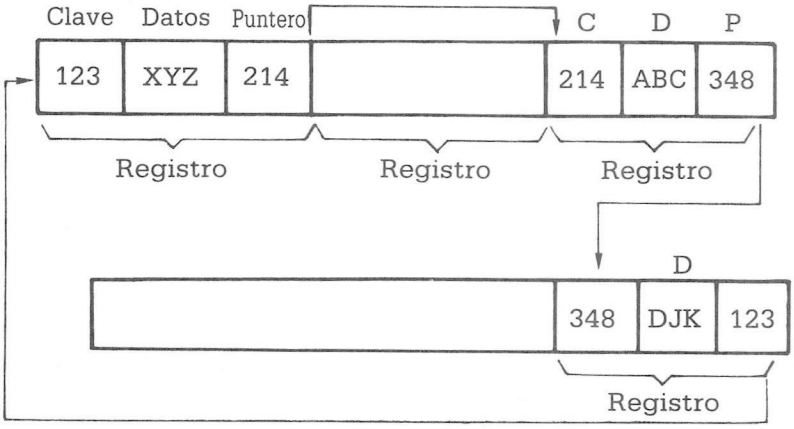


Figura 37.

Muchas veces esta estructura en anillo utiliza dos punteros, uno para relacionarse con el siguiente registro de la cadena y otro para relacionarse con el anterior, con lo cual la cadena se puede recorrer en los dos sentidos.

Se denomina *lista* a un conjunto ordenado de elementos. La técnica de listas usada con registros encadenados, se denomina de *"lista simple"*. En esta técnica, la cadena se fragmenta en varias partes, cada una de las cuales tiene su propio punto de entrada.

Sea por ejemplo la tabla siguiente de departamentos y empleados.

Departamento	Empleado
3200	27214
3300	32123
3400	00425
3500	82324
3600	01237

Figura 38.

Por cada código de departamento está el código del registro del primer empleado que forma la cadena de ese departamento. Por otra parte, los empleados podrán estar ordenados secuencialmente, encadenados secuencialmente o por número de empleado, mediante un segundo puntero.

En definitiva una lista simple es un indexado aplicado a una cadena a la que se le restringe la longitud a través del propio archivo.

Una variante es la *"lista invertida"*, que no usa el encadenamiento en los registros del archivo. Para ello se utiliza una entrada por cada registro en el archivo y listas con las referencias entre registros.

Los pasos a seguir son: acceso a la lista, acceso al índice principal y acceso al siguiente registro encadenado.

Las listas se pueden almacenar al principio del archivo en el mismo soporte o en otro diferente.

Un ejemplo puede ser el siguiente de un archivo de existencias.

Código Almacén	Código Producto
001	1348
	1427
	1532
	1985
002	3423
	0052
	4287
	5423
003	7235
	0138
	1532
	7235
004	9325
	0354
	0548
	0693
005	3427
.	5225
.	6834
.	0003
.	0101
.	.
.	.

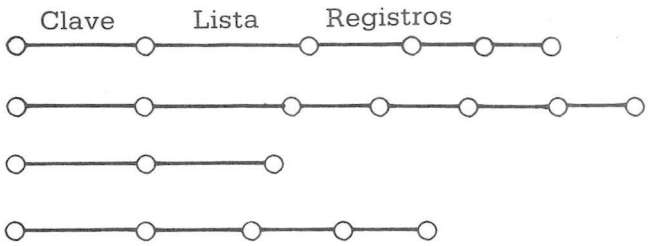
Figura 39.

En el archivo hay una entrada para cada registro de stock (representado por el código del producto). Los códigos de productos están en secuencia en cada proveedor. Los encadenamientos se hacen, no por punteros, sino mediante las listas. En la organización inversa se verá más a fondo su aplicación.

Las listas dan un tiempo de respuesta rápido, incluso menor que los archivos encadenados, pero requieren mucho más espacio de almacenamiento.

En resumen, las listas pueden esquematizarse de la siguiente forma.

Lista simple



Lista invertida

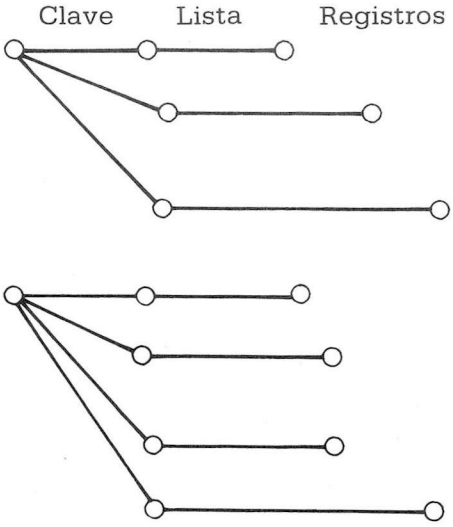


Figura 40.

7.– ORGANIZACION REGIONALIZADA

Utiliza como soporte físico el disco. Las claves de identificación se componen de un "prefijo de clave", llamado *radical*, de manera que el número de claves posibles de identificación tienen un valor M conocido. Vamos a ir desglosando todo esto. A cada radical posible de la clave, se le asigna un espacio del disco, llamado *región*. Dentro de la región, se situan los registros por orden de clave. Cuando queramos añadir o suprimir un registro, se accederá directamente a la región y a continuación se actualizará ésta de forma secuencial. Si el valor M es pequeño, podemos colocar los registros de la región sin mantener el orden correlativo de las claves. Para localizar una determinada clave, habrá que recorrer toda la región. El radical puede indicar una provincia, una sucursal, una familia de productos, etc.

8.– ORGANIZACION PARTICIONADA

Como su nombre indica, esta organización se basa en dividir o "particionar" el fichero en *módulos* de diferentes tamaños. Cada uno de estos módulos forman subficheros, identificados por un nombre. Este fichero se procesa por módulos completos. El soporte físico es el disco que lleva un Directorio encargado de contener cada uno de los nombres de los módulos y su situación física en el fichero, junto al tamaño total del módulo dado en la cantidad de registros físicos que lo componen.

Dentro de cada módulo, los registros se organizan secuencialmente. Podemos consultar y actualizar los registros de cada módulo, así como añadir o suprimir módulos enteros. Es claro que cuando ocurre esto, también hay que actualizar el Directorio.

Pongamos un ejemplo: Supongamos que necesitamos un fichero que contenga toda una serie de programas que podemos necesitar en cualquier momento. La organización particionada es ideal para construir nuestro fichero. Cada programa formará un módulo y sus instrucciones se colocarán secuencialmente en los registros. El directorio llevará el nombre de identificación de cada módulo, su situación física en el fichero y su tamaño. Con esta organización podremos formar verdaderas bibliotecas de programas.

9.– ORGANIZACION SECUENCIAL-INDEXADA

En este tipo de organización se crea en primer lugar, una distribución física de los registros en el fichero en forma secuencial y, a continuación, unas tablas o índices de manera que si se quiere acce-

der a uno de los registros del fichero, se le puede localizar de forma individual, tal como ocurría en la organización directa. Como vemos, en esta organización aparecen relacionados los conceptos secuenciales y directos, es decir, los registros están grabados en una secuencia ordenada y en el momento de la grabación, el método de acceso crea unas tablas o índices que hacen posible que el tratamiento lógico posterior pueda ser directo a un determinado registro, sin tener que pasar por los anteriores.

Debemos darnos cuenta de que, debido a la forma en que están físicamente grabados los registros, también se puede emplear un tratamiento lógico secuencial.

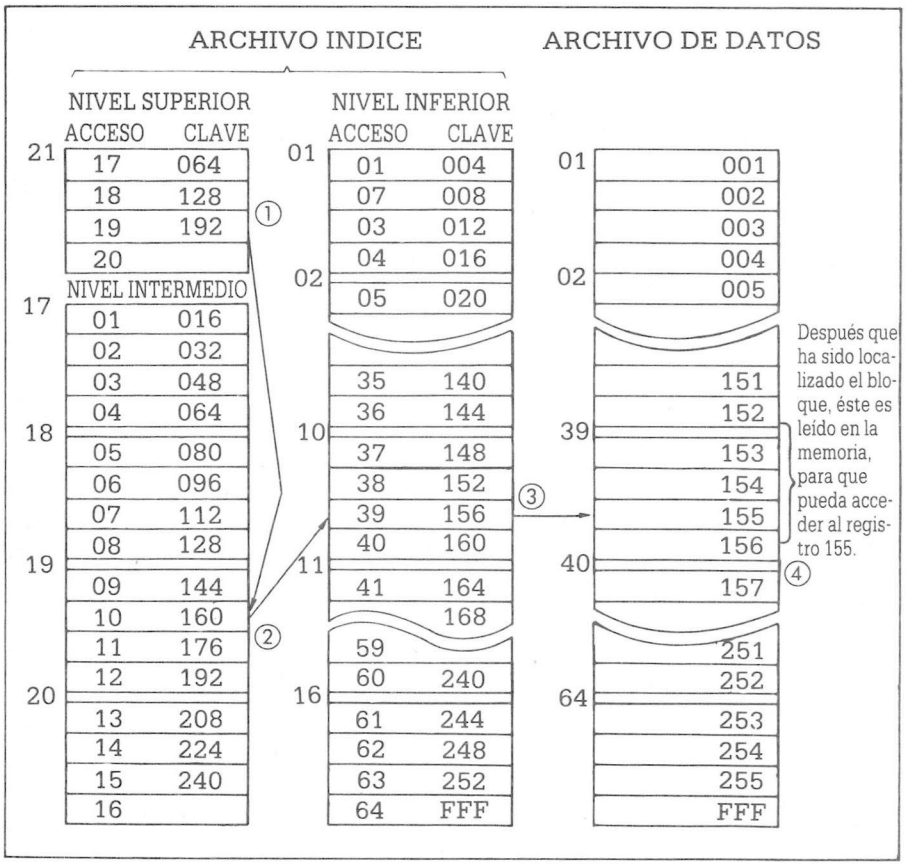


Figura 41.

¿Qué diferencia existe entre esta organización y la directa?. Fundamentalmente está en que en la secuencial-indexada la localiza-

ción directa del registro se hace a través de unos índices creados y actualizados por el método de acceso, mientras que en la organización directa, el proceso de localización se hace por la relación establecida previamente entre la clave y su situación física en el fichero. Esta relación la crea el programador.

Los ficheros organizados de forma secuencial-indexada, tienen que residir en dispositivos de acceso directo (disco, tambor, etc.).

Un ejemplo de esta organización puede ser un diccionario. Todas sus páginas comienzan por una palabra y esta palabra es la clave para localizar rápidamente las palabras que hay en el resto de la página.

Número de página	Palabra clave
1	ABACO
2	ABOLICION
3	ABSOLUTO
4	ACARICIAR
5	ACCIDENTAL
6	ACETILENO
7	ACORAZADO
8	ACUSADOR
9	ADAPTADOR
10	ADORNAR

Figura 42.

Si quiero encontrar la palabra Adivinar, me iré a la página 9, ya que la 10 comienza con ADORNAR.

10.– ORGANIZACION CON FICHERO INVERSO

En algún tipo de actividad puede ser aconsejable que aparezcan varios registros con la misma clave así como que un registro tenga distintas claves. A esto responde el "duplicado de clave" (varios registros con la misma clave) y el "multiclave" (un registro con varias claves).

Una organización secuencial-indexada puede manejar los duplicados de clave con facilidad, pero el multiclave es más complicado ya que se tendría que repetir el registro tantas veces como claves distintas tenga. El coste sería alto, con lo que se recurre a una nueva organización de ficheros, llamada organización inversa.

Dicha organización consta de dos partes:

a) Area primaria o fichero directo. En ella se graban los registros en cualquier orden.

b) Tabla de acceso, llamada fichero inverso, en la que para cada clave se presenta una lista que contenga las direcciones físicas de los registros que tenga dicha clave.

Creemos que un ejemplo muy clásico de esta organización va a aclararnos mucho todo lo expuesto. Tenemos una librería, en ella hay libros que tratan de Física, Química, Matemáticas, Informática, etc. Si pensamos un poco, sería lógico que más de uno de estos libros tratara sobre un mismo tema y que un libro abarque varios temas.

Vamos a organizar un fichero que nos ayude a localizar lo que deseemos saber sobre todas estas materias.

Las claves de acceso serían las materias que se exponen en los libros (Física, Química, etc.). Los registros contendrán los datos del libro (título, autor, editorial, etc.).

El fichero inverso o tabla de acceso estará formado por un índice en el que aparecerán los nombres de las materias, por orden alfabético, y para cada materia, los libros que hacen referencia a ella.

El área primaria, o fichero directo, la formarán los registros que se podrán colocar secuencialmente sin tener en cuenta las claves asociadas a cada uno de ellos.

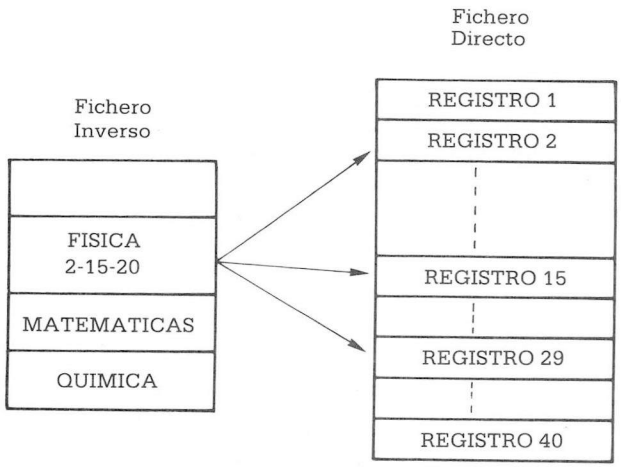


Figura 43.

El fichero inverso puede utilizar una organización directa con su consiguiente relación para el acceso, y se construirá simultáneamente con el directo. Normalmente se utiliza la organización se-

cuencial-indexada, y entonces lo que se hace es comenzar a crear un índice de claves secuencial a medida que salen los registros del fichero directo, para a continuación, mediante un proceso de clasificación, se ordenan las claves y se forma la organización secuencial-indexada del fichero inverso. Es importante que los registros que acompañen a cada clave, se ordenen en sentido creciente de sus situaciones físicas en medio del almacenamiento (normalmente disco), y así disminuir el tiempo de acceso a los registros con una misma clave.

En la figura 43 vemos un ejemplo de cómo se organizaría esto.

¿Qué ocurrirá cuando necesitemos modificar algún contenido de estos ficheros?. Si queremos modificar el contenido de un registro sin tocar el campo de la clave, sólo cambiará el fichero directo y no hará falta actuar sobre el inverso.

Si queremos añadir un nuevo registro al fichero directo, tendremos que actualizar el inverso con las claves que lleve asociadas el registro, y si apareciera una nueva clave, deberá insertarse en el lugar correspondiente del fichero inverso.

Cuando queramos suprimir un registro, no hay problema en cuanto al fichero directo: se elimina y se acabó; pero en el inverso hay que eliminar las claves que nos remitan al registro eliminado. Es evidente que si deseamos añadir, suprimir o modificar las claves, el fichero más afectado vuelve a ser el inverso, pues habrá que modificar todos los items afectados por la clave.

11.– CONCLUSIONES

De todos los tipos de organizaciones vistas, las más empleadas son: secuencial, directa, secuencial-indexada e inversa.

Si queremos realizar un fichero y estamos indecisos sobre qué tipo de organización debemos emplear, tendremos en cuenta lo siguiente:

Si el fichero sólo necesita ser procesado secuencialmente y no tiene que ser actualizado con asiduidad, emplearemos la organización secuencial.

Si necesitamos un fichero de tiempo de acceso muy pequeño y que deba ser procesado al azar, utilizaremos la organización directa.

La organización secuencial-indexada es ideal cuando se necesite procesar tanto de forma secuencial como directa.

Cuando un registro tenga más de una clave y se necesite un proceso directo, emplearemos la organización inversa.

Además de las organizaciones presentadas, que suelen ser standards en muchos fabricantes de Sistemas de Gestión de Ficheros, el programador puede diseñar otras, más o menos complejas, en función de sus necesidades y de su imaginación y creatividad, con el fin de reducir volúmenes de almacenamiento o tiempos de proceso.

Los factores que en definitiva afectan a la selección de la organización son: Volatilidad del archivo, actividad del archivo, tiempo de respuesta del sistema, diseño del sistema, tamaño del archivo, necesidad de reorganización y técnicas de recuperación y de respaldo.

Ficheros directos encadenados

Los registros se agrupan en apartados
bloques

cuando se quiere acceder a un registro
se le toma un apartado que se localiza
por su parte clave
cuando un apartado se llena, el
registro pasa al area de overflow
encadenandolo (osea el apartado se le
se encadenalmente)

- En la organización regionalizada cada región
se accede por un radical de la clave
- En la particionada hay un directorio
que nos da la situación de dos modulos
-

CUESTIONARIO CAPITULO 3

ORGANIZACION DE FICHEROS

1.– ¿Qué diferencias hay entre organización lógica y organización física?
2.– ¿Cuál es la principal característica de un fichero secuencial?
3.– ¿Para qué uso es adecuada la organización secuencial?
4.– ¿Qué diferencia una organización secuencial de la standard?
5.– ¿Para qué sirve el área de rebosamiento?
6.– ¿Qué es un archivo relativo?
7.– ¿Qué es un puntero?
8.– ¿En qué tipo de organización se utilizan los punteros?
9.– ¿Cómo se enlazan los registros entre el área principal y el área de rebosamiento?
10.– ¿Qué es un apartado?
11.– ¿Qué estructuras de archivos existen aparte de la lineal?
12.– ¿Cuántos punteros puede utilizar una estructura en anillo?
13.– ¿Qué se entiende por lista?
14.– ¿Qué es una lista invertida?
15.– ¿Qué son la región y el módulo?
16.– ¿En qué se parece un diccionario a la organización secuencial-indexada?
17.– ¿Qué es un registro multiclave?
18.– ¿Qué factores afectan a la selección de la organización?

CAPITULO IV

METODOS DE ACCESO A FICHEROS

1.– INTRODUCCION

Existen diferentes métodos de acceso a ficheros. Todos ellos tienen como misión organizar y mantener los ficheros de acuerdo con una determinada organización física, y facilitar su tratamiento lógico en una de las formas ya descritas en el capítulo anterior.

Los programas que sirven de intermediario entre el nivel lógico del usuario y el nivel de organización física del computador constituyen el llamado **Sistema de Gestión de Ficheros**, cuyas funciones principales son:

- Definir la organización del fichero.
- Definir la estructura de los registros.
- Crear el fichero almacenando los distintos registros, siguiendo las estructuras de los ficheros.
- Actualizar los ficheros mediante la modificación de campos, eliminación de registros y adición de otros nuevos.
- Consultar el fichero.

Vamos a estudiar de forma detallada, los principales métodos de acceso, que son:

1.– Secuencial (SAM).
2.– Directo (DAM).
3.– Secuencial-indexado (ISAM).
4.– Particionado (PAM).
5.– De memoria virtual (VSAM).

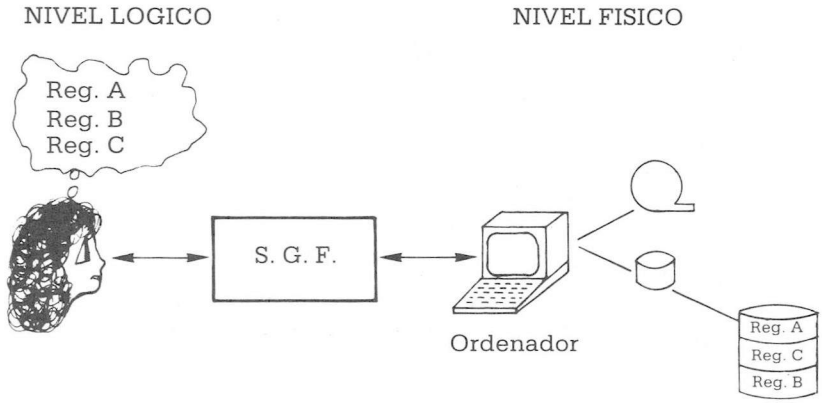


Figura 44.

Las siglas entre paréntesis que acompañan a cada uno de los cinco métodos, son las iniciales inglesas de cada uno de ellos y así se les conoce también.

2.– METODO DE ACCESO SECUENCIAL (SAM)

Con este método, se accede a los registros del fichero según el orden en que se almacenaron.

Las formas en que el fichero puede estar organizado son: secuencial, standard y en cadena. Independientemente del tipo de organización, el acceso secuencial obliga a comenzar por el primer registro del fichero y recorrerlo secuencialmente hasta llegar al último.

Existen diversas formas de efectuar el acceso secuencial. Veamos las tres más corrientes a partir de ejemplos prácticos de su uso.

a) Técnica de Análisis del contenido.

Supongamos que deseamos tener un fichero en el que podamos grabar los nombres de los contribuyentes de una Administración y el dinero que deben, de manera que podamos consultarlo y sacar mediante la impresora un listado de los "morosos". Por ejemplo, le decimos al ordenador "sacar la lista de los contribuyentes que deben más de 100.000 Pesetas" y en pocos segundos, la impresora nos la suministra. La forma de construir este fichero es usar la técnica de análisis de contenido. Mediante ella podemos acceder a cada uno de los registros del fichero, examinarlo y comprobar si cumple una determinada condición. Aquí no es importante la secuencia del fichero,

ya que basta con leerlo y ver qué datos cumplen la condición impuesta. En el ejemplo expuesto, la condición sería "deban más de 100.000 Pesetas".

El tipo de organización que se utiliza es la standard, aunque también se podría emplear la encadenada.

b) Padre/Hijo.

Supongamos ahora, que nuestra empresa se dedica a vender piezas y circuitos de recambio de videos y televisores en color. Deseamos conocer, mensualmente, el número de piezas que hemos vendido y cuántas quedan en el almacén.

Para procesar esto, vamos a emplear el proceso conocido como Padre/Hijo. Al final de cada mes en nuestro pequeño centro de cálculo, dispondremos de los siguientes elementos:

1) Un fichero maestro que contiene toda la información del mes anterior. Es el fichero llamado "Padre".
2) Un fichero que almacene la información sobre las ventas del mes que ha finalizado. Es el llamado "fichero de transacciones".
3) Un tercer fichero sobre el que se va a grabar toda la información necesaria para saber los nuevos valores y cuántas nos quedan en el almacén. Este fichero es el llamado "Hijo".

Es importante tener en cuenta que tanto los dos archivos maestros (padre e hijo) como el de transacciones, deben estar clasificados en un mismo orden según una clave. Esta clave puede ser el código asignado a cada pieza. Además los registros pueden llevar otros campos.

En nuestro ejemplo, los archivos maestros tendrían registros con los campos siguientes:

– Número asignado a cada pieza.
– Nombre de la pieza.
– Unidades vendidas ese mes.

En la figura vemos el esquema de este proceso.

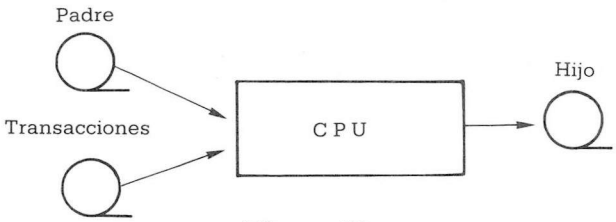


Figura 45.

Comienza el proceso y empiezan a pasar los registros del Padre y del de transacciones.

Clave	Datos
1	
2	
3	
4	
5	
6	
7	
8	

Padre

Clave	Datos
2	
4	
5	
7	
8	

~~Hijo~~ Transaciones

Figura 46.

Los números que aparecen son los correspondientes al número asignado a cada pieza.

Leemos el primer registro de estos dos ficheros. El del Padre es el número 1 y el del Hijo es el 2. Esto indica que en el mes no se ha retirado ninguna pieza con el número 1. Este registro se copia en el fichero Hijo sin ninguna modificación. Como el registro de transacciones no ha sido procesado, se conserva y se lee el siguiente registro del padre. Es el número 2 y se compara con el primero del de transacciones: 2 = 2. Hay proceso y se actualiza el Padre grabando la información en el Hijo. Ya no hay que conservar el 2 de transacciones.

Así seguirá haciendose el proceso y al finalizar, los registros del fichero Hijo quedarían así:

1	No actualizado
2	Actualizado
3	No actualizado
4	Actualizado
5	Actualizado
6	No actualizado
7	Actualizado
8	Actualizado

Este proceso se utiliza en ficheros de gran actividad y se pueden añadir o suprimir registros de una forma muy sencilla. Para añadir un registro en el maestro, el registro se sitúa en el fichero de transacciones con un código que indique que es una inserción.

Un defecto de este tipo de acceso es que su tiempo de proceso es alto, ya que no hay que olvidar que es del tipo secuencial.

c) Secuencial-Selectivo.

Sigamos hablando de nuestra empresa. Ahora vamos a organizar y procesar un fichero que nos diga el número de piezas de recambio de televisores en blanco y negro que vendemos y cuántas nos van quedando en el almacén. Ya es sabido que este tipo de televisores está desapareciendo del mercado por lo que las ventas de piezas serán mucho menores que en el caso de los videos y los televisores en color. Podríamos utilizar el acceso Padre/Hijo, pero en este caso no merece la pena porque las transacciones van a ser muchas menos.

Vamos a utilizar el acceso Secuencial-Selectivo, que sólo va a necesitar dos ficheros:

1) El fichero maestro, que va a contener la información almacenada hasta el mes anterior.
2) El fichero de transacción, con la información de las ventas del mes actual.

El fichero maestro se va a comportar como un fichero de Entrada/Salida y sólo se van a actualizar los registros que sufran un cambio en su información.

En la figura 47 representamos el esquema de este nuevo método de acceso.

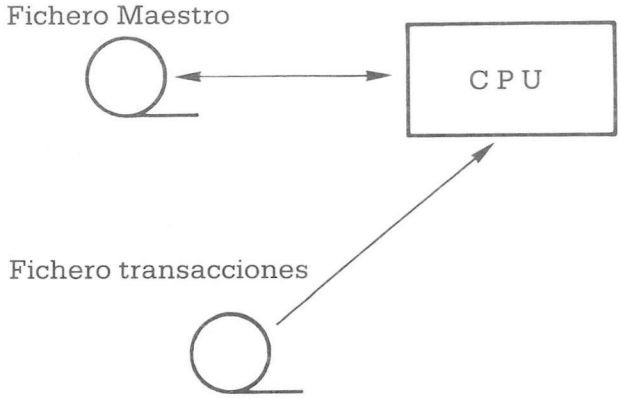


Figura 47.

Tanto el fichero maestro como el de transacción deben estar ordenados, tal como pasaba en el Padre/Hijo.

Los contenidos de los registros de ambos ficheros son los mismos que los del caso anterior. La forma de trabajo de este proceso es la siguiente:

Tanto los registros del maestro como los del tratamiento se procesan por bloques. Sólo los bloques del maestro cuyas claves de registros coincidan con las de los registros del de transacción, se modificarán. El medio de almacenamiento utilizado es el disco o el tambor magnético, ya que el fichero maestro se utiliza tanto como medio de entrada como medio de salida. Cuando el programa lea un bloque, debe conservar la pista donde está situado para que si se actualiza un registro, el ordenador pueda detectar la existencia del cambio y pueda volver a escribirlo en el mismo sitio. Una de las formas de ejercer este control de actividad es la utilización de un indicador, el cual se pone en ON cuando se actualiza un registro. Antes de que se lea un nuevo bloque, se examina el indicador. Si está en ON, se escribe el bloque y se pasa a OFF. Si el indicador está en OFF, no hay que reescribir el bloque, ya que no ha sufrido cambios.

Supongamos que nuestro maestro tiene bloques de 7 registros:

BLOQUE 1 →	1	2	3	4	5	6	7
2 →	8	9	10	11	12	13	14
3 →	15	16	17	18	19	20	21
4 →	22	23	24	25	26	27	28
5 →	29	30	31	32	33	34	35

Figura 48.

El de transacciones es el siguiente:

4	17	31	35

Figura 49.

Los bloques sin actividad (indicador en OFF) son el 2 y el 4.

Los bloques con actividad (indicador en ON) son el 1, 3 y 5.

Los registros con recuadro son los que han vuelto a ser escritos perfectamente actualizados.

¿Qué ocurriría si el fichero de transacción anterior fuera 4, 47, 31, 35?. Pues que la clave 47 no tiene igual en el fichero maestro, y el programa lo trata como un error y lo puede indicar en la pantalla o impresora del ordenador. Igual ocurriría en el método Padre/Hijo.

El tiempo de proceso es menor que en el caso del Padre/Hijo, ya que sólo se vuelven a escribir los bloques activos y el fichero maestro se puede emplear en otros procesos secuenciales. Con este método, no se pueden añadir o suprimir registros durante la actualización. Si fuera necesario hacerlo, hay que recurrir al acceso Padre/Hijo. Al hacer la actualización, la reescritura destruye la información acumulada anteriormente por lo que si se quiere conservar la información antigua, es necesario hacer una copia previa del fichero. Tampoco se pueden modificar los tamaños del bloque que se modifica, por lo que el espacio de reescritura no cambia.

Para terminar diremos que tanto el método secuencial-selectivo, como el Padre/Hijo, necesitan un solo acceso para cada registro del maestro, ya que para un registro del maestro las transacciones están juntas al haberse ordenado previamente por orden de código.

3.– ACCESO DIRECTO (DAM)

Para usar este método, la organización física de los registros tiene que ser la directa. Recordemos que en este tipo de organización debería existir un algoritmo o relación, perfectamente definido, entre la clave indicativa del registro y su ubicación física en medio del almacenamiento. Este medio será el disco magnético o el tambor magnético. Con este método se tarda, aproximadamente, el mismo tiempo para acceder a cada registro. Si los bloques del fichero poseen cada uno más de un registro, el acceso a cada bloque tiene la misma duración. Esta es ya una diferencia entre el acceso directo y el secuencial. En este último los tiempos de acceso a los registros no eran iguales.

El algoritmo de direccionamiento debería ser biunívoco, es decir, que para claves diferentes, debe de producir direcciones diferentes. Pero si existen muchas claves que no son utilizadas, la biunicidad obligaría a dejar mucho espacio desperdiciado en el almacenamiento. Para reducir este espacio se usan *"algoritmos de direccionamiento aleatorizado"*, que reducen el rango de las claves en función del porcentaje de claves que en realidad existen. Como consecuencia aparecen los *"sinónimos"* (dos o más registros cuyas claves aleatori-

zadas dan la misma dirección.).

Un algoritmo de aleatorización debe conseguir que:

- Cada clave se convierta en una dirección perteneciente al rango permitido.
- Se procure una distribución uniforme, con el fin de que existan pocos sinónimos.

Los sinónimos se almacenan en un área especial del fichero denominada *área de excedentes o rebosamiento.*

El porcentaje de posiciones que realmente se ocupan se llama *factor de empaquetamiento.* El factor de empaquetamiento es conveniente que esté entre el 80 y el 85%.

El primer registro que se graba en una posición se denomina *"registro original"* y los sinónimos que se graban en el área de excedentes se llamarán *"registros excedentarios"*.

Existen diferentes técnicas para el acceso directo según sea el tipo de algoritmo que se establezca. Vamos a estudiar a continuación algunas de esas técnicas.

a) Técnica de Direccionamiento directo.

Supongamos, de nuevo, que tenemos un fichero maestro y un fichero de transacciones, y vamos a actualizar el maestro con las informaciones grabadas en el de transacciones.

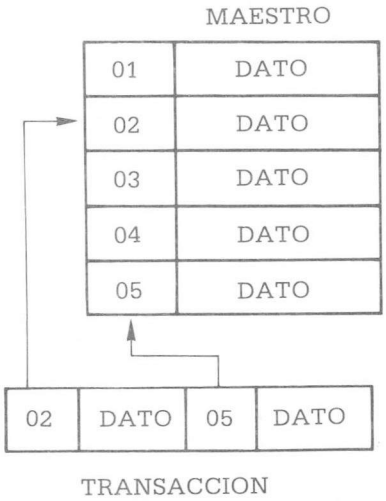


Figura 50.

Para realizar esto, el direccionamiento directo nos da dos posibilidades:

1) Dirección real.

Cada registro de transacción posee una clave que contiene la dirección de la pista o del sector donde se encuentra el registro maestro. Cuando el fichero se organiza almacenando un registro por bloque y además cada bloque se encuentra en un sector, la clave, como hemos dicho, nos lleva directamente al registro maestro.

En la figura 50 detallamos este tipo de acceso.

01, 02, ---, 05 indican los sectores del disco. Cada sector tiene un bloque y cada bloque un registro. Como vemos, la actualización se realiza empleando, en el de transacción, una clave con la dirección del sector.

¿Qué ocurre si el fichero se organiza de manera que cada bloque posea varios registros?. Ahora la clave tendrá dos campos. Un campo identifica al bloque y el otro al registro. Veamos un ejemplo de lo dicho.

REGISTRO			REGISTRO			REGISTRO			REGISTRO		
01	12	Dato	01	14	Dato	01	06	Dato	01	20	Dato
02	03	Dato	02	09	Dato	02	15	Dato	02	18	Dato
03	21	Dato	03	17	Dato	03	23	Dato	03	22	Dato
1.ª Clave	2.ª Clave										

Figura 51.

Lleva las dos claves y el campo de datos. Cuando se localiza el registro, se actualiza y luego el bloque se vuelve a escribir en el disco.

Tenemos tres bloques o sectores y cada bloque contiene cuatro registros. En cada registro los dos primeros campos son las claves. La primera de ellas nos indica el bloque y la segunda el registro. Cuando se accede a un bloque, el programa busca, de forma secuencial, el registro deseado. Es importante observar que los registros no están ordenados según la clave segunda. Eso está perfectamente permitido.

El fichero de transacción se podría representar así:

01	12	DATO	01	20	DATO	02	03	DATO	03	21	DATO

Figura 52.

2) Dirección relativa.

Este segundo método de direccionamiento directo accede a los registros del maestro utilizando una clave que indica la posición relativa del registro, sin especificar la dirección del sector del disco. Con el ejemplo de la figura, explicaremos este método.

SECTOR 1	01	DATO	02	DATO	03	DATO	04	DATO
SECTOR 2	05	DATO	06	DATO	07	DATO	08	DATO
SECTOR 3	09	DATO	10	DATO	11	DATO	12	DATO

TRANSACCIONES

10	DATO	02	DATO

Figura 53.

El de transacciones puede estar ordenado o no. Como siempre, el programa será el encargado de actualizar y reescribir los bloques en el disco.

En ambos métodos, la actualización lleva consigo destruir la información primitiva, por lo que si se quiere mantenerla hay que realizar copias del Maestro. Estas técnicas se pueden emplear también con ficheros standard y encadenados.

b) Direccionamiento por fórmula.

Este método se conoce también por el nombre "UCA" y con él se accede y procesa al fichero asignándole a cada registro maestro una clave única. La dirección del registro se obtiene aplicándole a la clave una fórmula o algoritmo. Es evidente que ese mismo algoritmo hay que aplicárselo a la clave del registro de transacción. En definiti-

va, va a ser el algoritmo el que va a determinar dónde se van a almacenar cada uno de los registros del Maestro.

¿Cuántas técnicas existen para este método?. Tantas como algoritmos utilice el usuario. Las claves no pueden ser alfabéticas, sólo númericas, ya que este método es matemático.

c) Técnica del resto de la división.

Es la más sencilla y suele dar buenos resultados.

En primer lugar se determina el número primo (M) más próximo y menor al número de direcciones del fichero. Para determinar la dirección de un registro se divide la clave por el número M y el resto de la división es la dirección relativa.

También se puede emplear otro número primo, pero se corre el riesgo de aumentar los sinónimos.

Supongamos que tenemos un fichero con ocho bloques. A cada bloque le vamos a asignar una posición relativa.

BLOQUE	BLOQUE	BLOQUE	BLOQUE	BLOQUE	BLOQUE	BLOQUE	BLOQUE
0	1	2	3	4	5	6	7

Figura 54.

O,1,2,---7 son las posiciones relativas de los bloques del fichero.

Vamos a emplear el método de "División por 7". Supongamos que vamos a almacenar estos 9 registros del maestro, los cuales tienen las siguientes claves: 014, 022, 104, 109, 019, 098, 099, 129, 016. Lo primero que hay que hacer es ir dividiendo las claves entre el número 7.

014 | 7 → 2 → [0]

022 | 7 → 3 → [1]

104 | 7 → 14 → 36 → [6]

109 | 7 → 15 → 39 → [4]

019 | 7 → 2 → [5]

098 | 7 → 14 → 28 → [2]

099 | 7 → 14 → 29 → [1]

129 | 7 → 18 → 59 → [3]

016 | 7 → 2 → [2]

Figura 55.

El "resto" de la división va a determinar en qué bloque relativo se va a almacenar el registro. El archivo maestro quedaría así:

014,098	022,099	016	129	109	019	104	
0	1	2	3	4	5	6	7

Figura 56.

Si observamos, los bloques relativos 1 y 2 tienen dos registros. Cuando existen dos o más registros ocupando un mismo bloque, se dice que estos registros son "sinónimos". El bloque relativo 7 no puede contener ningún registro ya que al dividir cualquier número entre 7, el resto nunca puede ser 7.

¿Para qué se puede utilizar este bloque?. La explicación es sencilla. Los métodos UCA pueden presentar problemas de "rebosamiento" o "excedentes", debido a que en un determinado bloque pueden existir tantos sinónimos que no haya espacio suficiente para almacenarlos. El bloque relativo 7, puede emplearse como área de excedentes. Parece lógico que la organización física del fichero sea la "encadenada" que resuelve con facilidad los problemas del rebosamiento.

El método empleado tiene la ventaja de utilizar un número primo que elimina muchos rebosamientos. Si hubiéramos dividido por dos, cinco de los registros (014, 098, 016, 104, 022) estarían en el bloque 0 y los restantes en el 1.

Los seis bloques que quedan serán áreas de rebosamiento.

Los registros de las transacciones no tienen que estar obligatoriamente ordenados. La actualización lleva consigo la destrucción de la información almacenada en los registros actualizados, por lo que si se quiere conservar hay que copiar el Maestro. Sólo hay que procesar los registros que se vayan a actualizar y su tiempo de acceso es bajo.

Para terminar, supongamos que tenemos grabado ya el fichero maestro.

MAESTRO

014 - 098	022 099	016	129	109	019	104	

Figura 57.

Vamos a actualizarlo con el siguiente fichero de transacciones.

014	016	042	109	019	104	022

Figura 58.

Los pasos que se dan son los siguientes:

1.°) La fórmula o algoritmo localiza el bloque que contiene el registro deseado.

2.°) Cuando se ha localizado el bloque, se escribe en la memoria y ya es el programa quien busca secuencialmente el registro a actualizar.

3.°) Se actualiza el registro y el bloque se vuelve a reescribir en la misma posición o sector del disco.

Si el programa repasa secuencialmente los bloques sin encontrar igualdad de claves, avisa que hay eror. Esos errores pueden aparecer en un fichero de errores.

En el ejemplo, los bloques recuadrados son los actualizados.

014 098	022 099	016	129	109	019	104

Figura 59.

El fichero de errores contendrá el registro de Transacción con la clave 042.

d) Técnica de plegado.

La clave se trunca, dando dos o más partes que se suman. El resultado se usa como dirección relativa.

De esta forma el número obtenido tiene un rango mucho menor que la clave.

Ejemplos con la clave 515034:

1° 51 + 50 + 34 = 135
2° 515 + 034 = 549
3° 553 + 104 = 657
etc.

e) Técnica del cambio de base.

La clave se cambia a una base de numeración diferente, desperdiciando los dígitos que sobren dentro del rango elegido.

Así, si cogemos un rango de direcciones de 1000 y queremos aleatorizar la clave 40132, usando por ejemplo la base 7, tendremos:

$6 \times 7^4 + 1 \times 7^2 + 3 \times 7 + 2 = 14478$

La dirección relativa será 4478.

f) Técnica del cuadrado.

El número de la clave se eleva al cuadrado y se utilizan los números centrales del producto como dirección relativa.

g) Técnica de truncamiento.

Se eliminan los dígitos del principio o del fin de la clave. Los que quedan dan la dirección del bloque.

Existen tantas otras técnicas como pueda imaginar el programador, pero tiene que seleccionarlas y probarlas con el fin de reducir los sinónimos.

Las ventajas del método son:

- Las transacciones no necesitan estar clasificadas.
- Los tiempos de respuestas son rápidos.
- Sólo se procesan los registros afectados.

Aunque también tiene desventajas como son:

- Puede haber muchos accesos al maestro si la actividad es alta y las transacciones no están clasificadas.
- El tiempo promedio puede ser alto si la actividad es grande.
- La reorganización del archivo requiere rutina específica.
- Hay que controlar los sinónimos.
- Es necesario copiar el maestro a fines de salvaguardarlos.

4.– METODO DE ACCESO SECUENCIAL-INDEXADO (ISAM)

Para utilizar este método la organización física del fichero debe ser secuencial-indexada. Recordemos lo explicado al tratar esta organización; se permite un tratamiento lógico de los registros en forma secuencial y en forma directa utilizando unas tablas de referencia, llamadas índices, que forman parte del fichero. La misión de estos índices es relacionar las claves indicativas de algunos registros, con su localización física dentro del volumen en el que están grabados.

El fichero va a estar compuesto por las tres áreas siguientes:

– Area primaria de datos.
– Area de índices.
– Area de excedentes.

Vamos a seguir, con mucha atención, el proceso de creación y actualización de un fichero de este tipo, que nos ayudará a comprender el contenido y la función de cada una de las áreas o zonas que forman el fichero. Recordemos que el fichero se tiene que crear, obligatoriamente, de forma secuencial, siguiendo el orden de secuencia ascendente.

Comencemos paso a paso el proceso:

1.°) Vamos a reservar el espacio necesario para cada una de las tres áreas anteriores, es decir, dividimos el fichero en tres partes, de acuerdo al tamaño que pensamos va a tener.

2.°) Comenzamos a grabar los registros. Lo haremos comenzando por el de la clave indicativa más baja, que quedará grabado al principio de la pista 1 del primer cilindro de los reservados para el área primaria de los datos. A continuación, y sobre la misma pista 1, se van grabando los registros en secuencia ascendente, hasta completarla.

3.°) Cuando hemos completado la pista 1, no pasamos a la siguiente, sino que empezamos a crear una tabla, llamada índices de pista. Esta tabla va a contener la clave indicativa del último registro que se haya grabado en la pista que se ha completado. Esta clave será la mayor de todos los registros contenidos en la pista. Este índice se va a grabar sobre la pista 0 del cilindro.

Una vez grabada en la tabla la clave correspondiente a la pista 1, continuamos grabando sobre la pista 2. Cuando la completemos, se crea una nueva clave en la tabla, que corresponderá a la mayor de los registros de la pista 2.

Seguiremos así, grabando sobre las pistas de este primer cilindro. El índice de pistas va a ir adquiriendo la forma de la figura:

N.° de Pista	Clave indicativa del último registro
1	100
2	150
3	300
4	325
⋮	⋮

Figura 60.

¿Qué ocurre cuando llenamos todas las pistas de este primer cilindro?. Pasamos al siguiente paso de nuestro proceso, que corresponde al cuarto.

4.°) Tenemos que crear otra tabla, que ahora llamaremos índice de cilindros. Va a estar formada por la clave del último registro grabado en el cilindro y va a ser la clave más alta de todos los registros contenidos en el mismo. El área dedicada a esta tabla, para su grabación, será la correspondiente a la reservada a la zona de índices.

5.°) El siguiente registro en secuencia a este último, lo vamos a grabar sobre la pista 1 del segundo cilindro, en su comienzo, y así continuamos rellenando la pista y la nueva tabla de índices de pista correspondiente a este cilindro.

6.°) Al terminar este segundo cilindro, se rellena el correspondiente índice de cilindros y así continuamos.

7.°) Si este índice de cilindros llegara a ser muy grande, podemos crear otro nivel de índices, llamado índice Maestro, que se grabará en el área de índices.

8.°) La estructura final del fichero sería la siguiente:

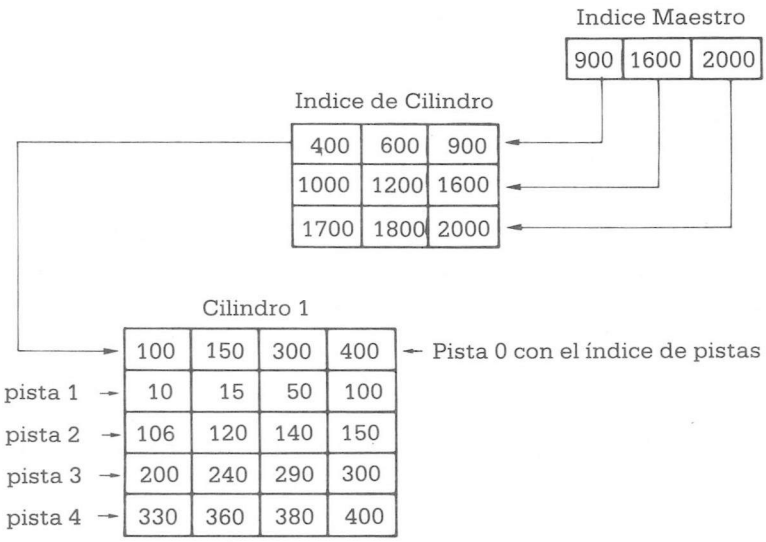


Figura 61.

Debe quedar muy claro que estos índices son creados y mantenidos por el método de acceso secuencial-indexado de forma automática y sin intervención del operador del ordenador.

Si el fichero de transacciones dice que hay que actualizar el registro maestro de clave 140, los pasos a seguir por el método de acceso serían:

a) Ver en el índice maestro la clave inmediatamente superior a 140. Sería la 900.
b) En el índice de cilindros cogería la clave correspondiente a la inmediatamente superior a 140. Sería la 400.
c) Pasa al índice de pistas y coge la clave inmediatamente superior a 140. Sería la 150.
d) Pasa a la pista n.° 2. La carga en memoria, la lee hasta llegar al registro de clave 140 y procede a la actualización.
e) Vuelve a escribir la pista número dos.

Si quisiéramos actualizar el registro de clave 118 es fácil adivinar que daría error. Siga el procedimiento y lo comprenderá.

¿Cómo resolveríamos la adición del registro de clave 118 en el maestro?. Lo que se hace en este caso es añadir este registro en el lugar que le corresponde en secuencia, entre el registro 106 y 120. A continuación se corren hacia atrás todos los registros de la pista afectada, con lo que habrá alguno que no quepa físicamente en dicha pista (el 150), y ese se graba en el área de Excedentes, encargándose el método de acceso de mantener unos apuntadores que lo permitan localizar.

Como vemos, esta estructura tiene un acceso directo a través de los índices y un acceso secuencial al ir leyendo los registros en las pistas.

No es deseable que el número de adiciones sea grande ya que la localización de registros a través de índices y apuntadores es lenta, con el consiguiente consumo de tiempo de ordenador.

También es necesaria un área de excedentes, que se puede hacer de dos maneras:

a) Area de excedentes única. Se reservan varias pistas contiguas para todos los excedentes. Adecuada para una distribución uniforme de los excedentes. Ahorra bastante espacio aunque la búsqueda de registros excedentarios es muy lenta.

b) Area de excedentes por cilindros. Se reserva una o más pistas en cada cilindro con el fin de almacenar los excedentes de ese cilindro. Se desperdicia más espacio que en el caso anterior.

Hay sistemas operativos que permiten simultáneamente los dos métodos con el fin de unir las ventajas y eliminar los inconvenientes.

En general los registros excedentarios no se bloquean nunca y

llevan punteros con el fin de encadenar en la secuencia correcta los registros.

5.– METODO DE ACCESO PARTICIONADO (PAM)

También conocido con el nombre PAM (siglas inglesas del método). La organización física utilizada por el fichero será la Particionada. Recordando lo visto ya en el capítulo anterior, esta organización se basa en que los registros están agrupados por módulos, es decir, el fichero se "parte" en varios trozos. Cada trozo es un módulo y cada módulo es reconocido por un nombre. Un área reservada al principio del fichero, llamada Directorio, va a almacenar los nombres de los módulos. Cada entrada del directorio constará del nombre del módulo y su dirección física de comienzo dentro del medio de acceso directo, único válido como soporte que admite esta organización. Los módulos se van grabando uno detrás de otro según vayan llegando.

En la figura tenemos un fichero con tres módulos y su correspondiente Directorio.

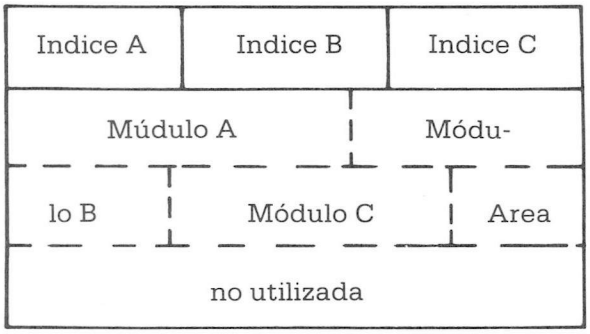


Figura 62.

Después de leer el párrafo anterior, seguro que el lector se habrá dado cuenta de que el acceso a este fichero se va a componer de un acceso directo a través del Directorio y de un acceso secuencial para llegar a un determinado registro del módulo.

Si queremos insertar nuevos módulos al fichero se podrá hacer a continuación del último y haciendo la entrada correspondiente en el directorio, o bien, si tenemos espacio suficiente, sobre el área dejada por un módulo que haya sido borrado.

Podríamos decir que un fichero particionado está compuesto de varios ficheros secuenciales.

Veamos un ejemplo de cómo se accede a este tipo de ficheros. Volvemos sobre el ejemplo expuesto, al hablar de esta organización. Queremos archivar una serie de programas. A estos programas hay que darles un nombre. Supongamos que tenemos cuatro programas cuyos nombres son: NEMO, YELLOW, BETA y ALFA.

El fichero quedaría organizado así:

NEMO	YELLOW	BETA	ALFA
0001	0400	0800	1000
NEMO			
YELLOW			
BETA			
ALFA			

Figura 63.

0001 = Dirección de comienzo de NEMO
0400 = Dirección de comienzo de YELLOW
0800 = Dirección de comienzo de BETA
1000 = Dirección de comienzo de ALFA

Queremos acceder al programa BETA. Como siempre será el método de acceso el encargado de hacerlo. Primero accederá al Directorio y localizará el índice "BETA 0800". Si lo tenemos grabado en Disco, moverá la cabeza de lectura/grabación hasta la dirección física 0800 y comenzará a leer el programa y a cargarlo en la memoria del ordenador.

Llegará a leer hasta la dirección 0999, ya que la 1000 corresponde al programa ALFA.

Observemos que cuando el programa sea procesado, el método no vuelve a grabarlo en el fichero ya que, en este caso, no hay que actualizar ningún registro, se trata de un programa y la lectura sobre disco no es destructiva. Sólo en el caso de que una vez procesado BETA, no nos interese mantenerlo en el fichero y deseemos reemplazarlo por otro, al que llamaremos GAMMA, de igual tamaño que el BETA, el método actualizará el nombre en el directorio y lo grabará a partir de la dirección 0800.

6.– METODO DE ACCESO DE MEMORIA VIRTUAL (VSAM)

Conocido con el nombre de VSAM, que son las siglas inglesas de este acceso, es el método utilizado en los equipos, que disponen de memoria virtual. Como su organización física no fué estudiada en el capítulo anterior, lo vamos a hacer ahora.

Existen tres posibles organizaciones para soportar estos ficheros:

- Ficheros en secuencia de Entrada (ESDS), equivalentes a ficheros secuenciales.
- Ficheros en secuencia de clave (KSDS), equivalentes a ficheros secuenciales indexados.
- Ficheros de registro relativo (RRDS), equivalentes a los ficheros con organización directa.

Antes de comenzar la explicación de estas organizaciones, vamos a definir una serie de conceptos nuevos que van a ser comunes a los tres.

1.°) *Intervalo de control.* Es la nueva unidad de transmisión de información entre la memoria auxiliar y la memoria principal utilizada por este nuevo método VSAM y que se compone de un número entero de registros físicos o bloques. Es obligatorio que este intervalo de control tenga una longitud fija que debe ser un múltiplo de 512 bytes. Este intervalo de control estará compuesto de registros lógicos, registros físicos o bloques, un área libre reservada por si en el futuro hay que añadir información al fichero y finalmente, de unos caracteres de información que van a indicar cosas tales como cuál es el nivel de ocupación del intervalo de control. Vamos a representar en la figura, la constitución del intervalo de control.

Intervalo de Control					
Bloque 1		Bloque 2	Bloque 3	Bloque 4	
Reg. lógico 1	Reg. lógico 2	Reg. lógico 3	Reg. lógico 4	Area LIBRE	INFORMACION

Figura 64.

Los bloques o registros físicos contienen registros lógicos cuya longitud es variable y pueden ser expandidos.

2.º) *Area de Control.* Los intervalos de control se van a agrupar a su vez en las llamadas áreas de control. Cada área de control de un mismo fichero, debe tener el mismo número de intervalos de control. El fin que se persigue al incluir estas áreas de control, es poder definir el espacio libre dentro del fichero. Este espacio libre será un porcentaje de los intervalos de control que comprenden cada área de control. Supongamos que definimos un fichero que va a tener, en cada área de control, ocho intervalos y vamos a definir también el espacio libre de cada área, como el 25% de esos 8 intervalos, es decir, al cargar el fichero, se ocuparán seis intervalos de control y se dejarán dos de cada área como espacio libre. Se dejan dos porque es el 25% de 8. Un área de control de este fichero, tendrá una organización como la de la figura. Cada intervalo tendrá sus registros físicos, lógicos y el área libre y de información. Sólo dos de ellos estarán completamente vacíos.

AREA DE CONTROL

INTERVALO LLENO
INTERVALO LLENO
INTERVALO LLENO
INTERVALO LLENO
INTERVALO LLENO
INTERVALO LLENO
INTERVALO VACIO
INTERVALO VACIO

Figura 65.

Estos son los nuevos conceptos introducidos por esta organización y que como ya hemos dicho, son comunes a los tres tipos de ficheros tratados por el método VSAM.

1.– Ficheros en secuencia de Entrada.

Son ficheros ESDS y recordemos que son equivalentes a los se-

cuenciales. Los registros se van grabando en los intervalos de control en el mismo orden en que van llegando, sin dejar ningún espacio libre en los intervalos de control. Sí dejará espacio libre en el área de control, tal como hemos dicho al definirla. La adición posterior de nuevos registros se hará siempre a continuación del último grabado. Su proceso es siempre secuencial. Si el usuario de este fichero mantiene, porque le interesa, unos índices indicativos del lugar donde ha grabado el método de acceso VSAM cada registro, se podría hacer posteriormente un acceso directo por la dirección del registro. Esta dirección se la da al usuario el método de acceso VSAM. Para terminar diremos que este tipo de ficheros admite registros de longitud fija o variable.

2.– Ficheros en secuencia de clave.

También llamados KSDS, son equivalentes al secuencial con índices (secuencial-indexado). El proceso de carga o grabación es muy parecido al ya descrito para los ficheros secuenciales-indexados, es decir, grabando en secuencia ascendente por la clave de cada registro y construyendo los índices de referencia. Las diferencias están en que en este caso los registros se van grabando en los intervalos de control, pero siempre dejando los espacios libres correspondientes dentro de cada intervalo de control y del área de control. Observemos, ahora, que mientras en el ESDS no se podía dejar espacio libre en los intervalos, sólo en el área de control, en el KSDS sí se dejan, tanto en el intervalo como en el área de control. Por otro lado, en el caso del KSDS cada entrada del índice de primer nivel apunta a un intervalo de control y contiene la clave del último registro grabado dentro del mismo (clave más alta) junto con la dirección de comienzo de dicho intervalo de control. Esta dirección viene indicada en términos de número de bytes relativo desde el comienzo del fichero (RBA).

Todos los índices de los intervalos de control de una determinada área de control, van dentro de un intervalo de control de los del área de índices de primer nivel. Se pueden constituir otros niveles siguiendo siempre el mismo criterio. La figura 66 resume todo lo dicho.

En esta figura vemos que el Area de índices tiene tres niveles. El segundo nivel nos indica ya el número de áreas de control del fichero que en este caso serían cuatro (nos falta la representación del área correspondiente a la dirección 2000). El tercer nivel nos indica el número de intervalos de control, para cada área. Observamos que cada área tiene tres intervalos de control, estando llenos dos de ellos y

uno libre, es decir, el porcentaje de área libre sería del 33,3% del área de control. Finalmente añadiremos que cada intervalo tiene sus bloques con la correspondiente área libre y los bloques de información.

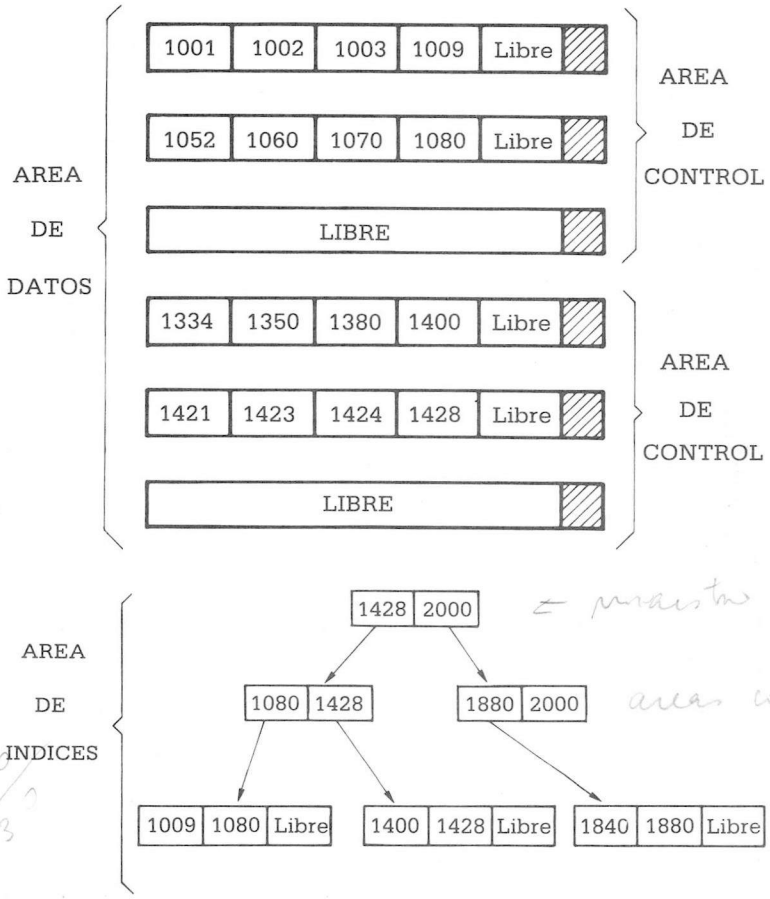


Figura 66.

La forma de acceso es típica del secuencial-indexado, directo para los índices y secuencial para los intervalos de control, y a su vez, dentro de estos dos tipos de acceso, puede ser por la clave del registro o por la dirección relativa (RBA), ya que como se ha dicho anteriormente, el propio método de acceso VSAM devuelve al usuario, después de la grabación, la dirección relativa (RBA) correspondiente.

Recibe el nombre de "CLUTER" el conjunto formado por las Areas de datos y de índices. El método de acceso VSAM mantiene y

utiliza catálogos con la información referente a los distintos "CLUTER" de una determinada instalación de informática. Esto facilita y agiliza su manejo, localización, etc.

Cuando haya que añadir más registros al fichero, después de haberlo grabado, la forma de hacerlo será situarlos en el lugar que les corresponda, dentro de su secuencia de clave aprovechando los espacios libres y actualizando los índices correspondientes. Recordemos que va a ser el método de acceso VSAM el que se va a encargar de realizar estas inserciones en el fichero. De esta manera se evita la utilización del Area de excedentes de los ficheros Secuenciales-Indexados y el tener que reorganizar el fichero.

Este tipo de ficheros KSDS admite tanto registros de longitud fija como variable.

Adicionalmente a la estructura primaria de índices explicada, el método de acceso VSAM tiene la posibilidad de construir índices alternativos que son unas estructuras que permiten la entrada al fichero y la consiguiente localización de cada registro a través de un dato que es diferente a su clave primaria. Supongamos que queremos acceder a un fichero para recabar información del comportamiento de distintos materiales en función de la temperatura. Con los índices alternativos se podría acceder a dicha información por el número del catálogo del material, por su nombre, etc. La utilidad de estos índices estriba en que así se evita la necesidad de mantener varias copias de la misma información organizadas de forma diferente, para uso de distintas aplicaciones.

3.– Ficheros de Registro Relativo.

Llamados RRDS y equivalentes a ficheros con organización directa. Admiten únicamente registros de longitud fija. La forma de localizar los registros es por su número de registro relativo respecto al comienzo del fichero. Para explicar esto mejor, diremos que a cada registro de los que van a ser grabados se le asigna un número de registro relativo, es decir, se le asigna un "hueco" donde se va a situar dentro del fichero. Es importante detenerse aquí y hacer hincapié en que la asignación de cada registro se hace antes de la grabación. Puede ocurrir que en la carga inicial algunos de los registros no se graben y en el fichero quedarán vacios sus "huecos". En la figura vemos como quedaría un intervalo de control.

Figura 67.

Si queremos añadir el registro 22, por ejemplo, éste irá a parar a su hueco entre el 21 y 23 y si es el 24 irá a un hueco entre el 23 y el 25. Como la longitud de los registros es fija y se sabe de antemano el número de ellos que entran en cada intervalo de control, como su número de registro relativo, el acceso directo a un registro será muy rápido. También se puede acceder al fichero de forma secuencial.

7.– ELECCION DEL METODO DE ACCESO

No existen reglas absolutas para la elección de un determinado método de acceso (de los explicados en este capítulo) cuando se vaya a construir un fichero. Todo queda en manos del programador, operador y usuario del fichero, pero es muy conveniente tener en cuenta una serie de aspectos o criterios que pueden aportar ideas sobre cuál puede ser el mejor método.

Vamos a exponer una serie de estos criterios:

El primero a tener en cuenta es el dispositivo físico sobre el que se va a almacenar el fichero. Evidentemente si el soporte va a ser la cinta magnética, o la tarjeta perforada, la organización y el método de acceso del fichero será secuencial.

¿Qué ocurre si el soporte es el disco magnético?. Pues que aparecen distintas posibilidades y además hay que tener en cuenta diversos factores:

a) Actividad del fichero.

Para actividades bajas está más indicado el acceso directo y por lo tanto, los métodos DAM, ISAM, VSAM-KSDS o VSAM-RRDS.

Cuando la actividad crece por encima del 60%, es preferible el acceso secuencial y por tanto los métodos SAM, ISAM, VSAM-ESDS o VSAM-KSDS.

b) Volatilidad del fichero.

Con baja volatilidad (pocas inserciones o borrados), se puede uti-

lizar el ISAM. Cuando la volatilidad crece, aumentan las necesidades de reorganizar el ISAM al llenarse el Area de excedentes, y es mejor utilizar cualquiera de los tres tipos VSAM.

c) Tamaño del fichero.

El tamaño impone una serie de restricciones que vamos a analizar a continuación.

1) Para ficheros DAM o ISAM, es obligatorio que todos los paquetes de discos que contengan el fichero, estén montados durante el proceso.
2) Para los ficheros VSAM, a los que se accedan secuencialmente, sólo tienen que estar montados los volúmenes en que residan los registros que van a ser accedidos. Si queremos acceder secuencialmente a los registros que van del 30.000 al 40.000, sólo montaremos el volumen que los contenga.
3) Para ficheros SAM sólo hay que montar, en cada momento, el volumen donde residan los registros que se van a tratar. Es muy normal que el fichero SAM ocupe más de una cinta magnética. En este caso se van montando una detrás de otra, según las vaya necesitando el proceso.

d) Tiempo de respuesta a las consultas.

En la actualidad, muchas aplicaciones informáticas requieren tiempos de respuesta muy bajos. Pensemos en un servicio de reservas de billetes, atención a los clientes de una sucursal bancaria, etc. Ese tiempo de respuesta se invierte en acceder a un determinado fichero, por lo que es importante localizar los registros lo más rápidamente posible. Estos casos piden organizaciones que permitan el acceso directo DAM, ISAM o VSAM y, entre ellas, seleccionar las que mejor satisfagan los objetivos de tiempo mínimo.

e) Tipo de tratamiento.

Si necesitamos un fichero que pueda ser tratado y accedido tanto de forma secuencial como directa, habrá que elegir una organización que lo permita. Organizaciones típicas son ISAM o VSAM.

Por último, presentaremos en una tabla los tipos de acceso que son posibles para cada una de las operaciones básicas y para las organizaciones de memoria no virtual.

Operación / Organización	Creación	Actualización	Consulta
Secuencial	Secuencial	Secuencial	Secuencial
Secuencial indexada	Secuencial	Secuencial/ directo	Directo
Directo	Directo	Directo	Directo
Fichero inverso	Sec/directo	Sec/directo	Directo

CUESTIONARIO CAPITULO 4

METODOS DE ACCESO A FICHEROS

1.– ¿Qué funciones realiza un Sistema de Gestión de Ficheros?
2.– ¿Cómo accede a los registros el método SAM?
3.– ¿Qué organizaciones de ficheros permiten el método de acceso secuencial?
4.– ¿Cuántos ficheros son necesarios en un proceso padre-hijo?
5.– ¿Cuántos ficheros necesita un proceso secuencial-selectivo?
6.– ¿Cuáles son las posibilidades del direccionamiento directo?
7.– ¿Qué es el método UCA?
8.– ¿Qué tipos de técnicas se usan en el direccionamiento por fórmula?
9.– ¿Qué áreas forman un fichero ISAM?
10.– ¿Qué debe hacer el usuario para mantener los índices?
11.– ¿Cómo se construye el área de excedentes?
12.– ¿En dónde se almacenan los nombres de los módulos en un fichero PAM?
13.– ¿Qué es un intervalo de control?
14.– ¿Qué es un fichero KSDS?
15.– ¿A qué se llama Cluter?
16.– ¿Qué método de acceso utilizaría si la volatilidad del fichero es baja?

SEGUNDA PARTE

BASES DE DATOS

CAPITULO V

CONCEPTOS FUNDAMENTALES SOBRE BASES DE DATOS

1.– ORIGENES HISTORICOS

Para comprender el nacimiento y evolución de los sistemas de bases de datos, es necesario conocer el medio informático de los últimos años. Las aplicaciones, que necesitaban de la utilización de ficheros de datos, operaban únicamente con sus propios ficheros. El único caso en que varias aplicaciones utilizaban ficheros comunes era cuando una de ellas creaba el fichero y lo utilizaba para consultas, actualizaciones y cualquier otra operación que necesitara realizar y, posteriormente y de forma independiente, una segunda aplicación utilizaba el mismo fichero como entrada de datos. Es decir, un fichero siempre tenía una aplicación dueña y, como mucho, era "prestado" a otras aplicaciones que tan solo podían accederlo para operaciones de extracción de información y nunca modificaban su contenido.

Por otro lado, los sistemas operativos no disponian de multiprogramación, es decir, el ordenador sólo podía ejecutar en un instante dado un único programa y por lo tanto, la seguridad de un fichero estaba garantizada sin más que los programas que lo manejaran realizasen las operaciones correctamente. De alguna forma se puede afirmar que el responsable de la seguridad de un fichero era el programa que lo actualizaba e, indirectamente, el usuario que utilizaba dicho programa.

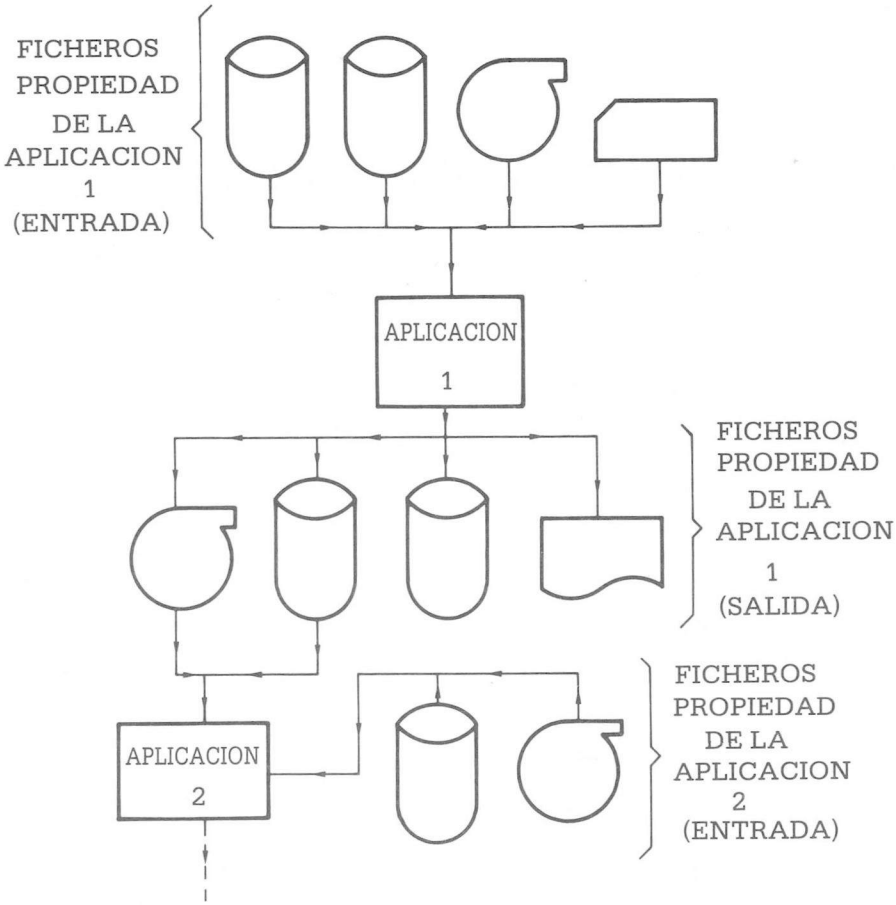


Figura 68. Antes de la aparición de las bases de datos, cada aplicación era propietaria de ciertos ficheros de datos que eran actualizados exclusivamente por ella. Como mucho podían ser "prestados" a otra aplicación, pero el término propiedad era siempre muy rígido.

Al ampliarse las aplicaciones y desarrollarse otras nuevas muy ligadas a los mismos ficheros utilizados por las aplicaciones antiguas y al implementarse sistemas operativos muy potentes que permitían la multiprogramación, es decir, que en el mismo instante se podían ejecutar distintos programas que actuaran con los mismos ficheros, aumentaron notoriamente los problemas para la gestión de los tradicionales ficheros. Por un lado las características exigidas al fichero variaban según la aplicación que lo utilizaba y por otro lado, al poder ejecutar simultáneamente dos programas que actualizaran un mismo fichero, las operaciones que ejecutaba uno podían y solían

ser incompatibles con las que necesitaba el otro.

La situación llegó a un punto insostenible dado que ninguna de las dos opciones a elegir resultaba satisfactoria. Si se optaba por mantener ficheros independientes para cada aplicación, resultaba que ciertos datos debían estar repetidos en muchos ficheros, con lo que se utilizaba mucha más memoria de almacenamiento de la estrictamente necesaria y por otra parte, al tener copias del mismo dato en más de un fichero, suponía la necesidad de efectuar múltiples actualizaciones cada vez que se quisiera modificar, con lo que dificultaba mucho la labor de mantenimiento. La otra solución era igualmente mala ya que si se unificaban todos los ficheros similares de las distintas aplicaciones en un único fichero, se llegaba a conflictos irresolubles en cuanto a su utilización conjunta.

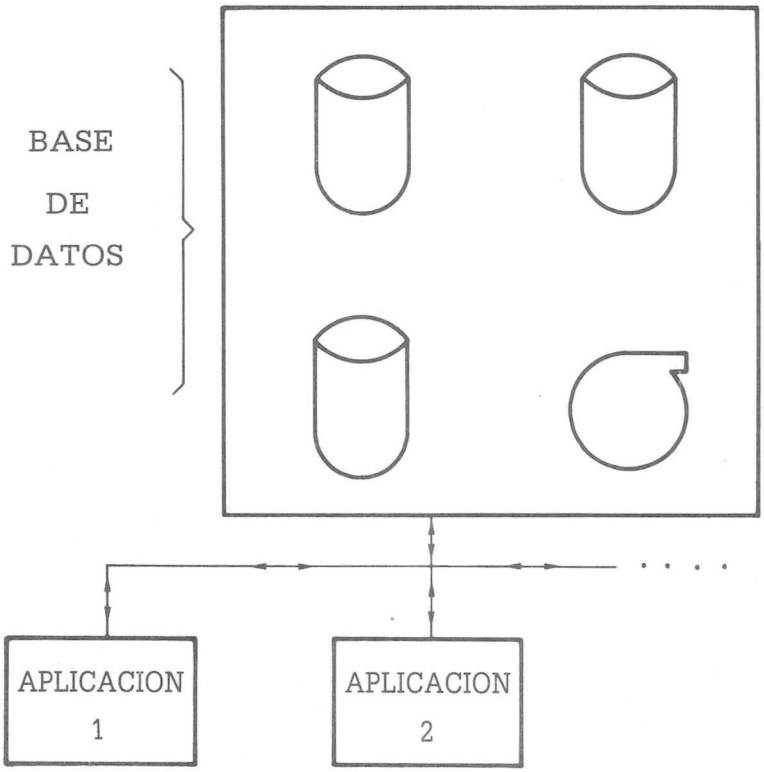


Figura 69. Con la aparición de las bases de datos, los ficheros dejan de ser propiedad de las aplicaciones pasando a ser compartidos por todos los usuarios

En casi todos los sectores de actividad humana, cuando se llega a un punto no satisfactorio, se concentran los esfuerzos de investigación hasta que el problema es obviado. En este caso, el resultado de esos esfuerzos fué un producto altamente novedoso y que supuso un gran salto en la informática de gestión: los sistemas de bases de datos.

2.– SISTEMAS DE BASES DE DATOS

Cuando en el principio de la década de los sesenta se puso de moda el término base de datos, muchos centros de cálculo se limitaron a cambiar de nombre a sus archivos y en vez de seguir llamándolos ficheros, comenzaron a llamarles bases de datos. Evidentemente este cambio solo de nombre no aportaba más ventajas que una simple mejora de imagen.

Para que un fichero puede dejar de denominarse como tal y pase a ser una base de datos, se deben verificar algunas condiciones obligatoriamente. Estas condiciones son las que caracterizan totalmente a las bases de datos y pueden ser consideradas como parte de su definición.

2.1 Definición de base de datos.

Se llama base de datos, a un conjunto de datos relacionados entre sí que se encuentran almacenados en una única colección, sin redundancias innecesarias y que cumplen las siguientes condiciones:

1.– Los datos están almacenados en diveros soportes de información de tal forma que son independientes de los programas que los manejan.

2.– Su utilización no está restringida a una única aplicación, siendo posible su acceso por varias aplicaciones, incluso simultaneamente.

3.– Para gestionar la información contenida en la base de datos, es decir, para incluir nuevos datos, borrar datos ya existentes, o modificarlos, se emplean procedimientos especialmente diseñados para optimizar el funcionamiento del sistema. Estos procedimientos no solo se encargan de la actualización, sino que también permiten la obtención de datos para su utilización mediante programas.

2.2 Sistema de bases de datos.

Cuando varias bases de datos, totalmente independientes se-

gún sus diseños, se reunen para facilitar su manejo al realizarlo de forma integral, se dice que el conjunto de todas las bases reunidas, es un sistema de bases de datos.

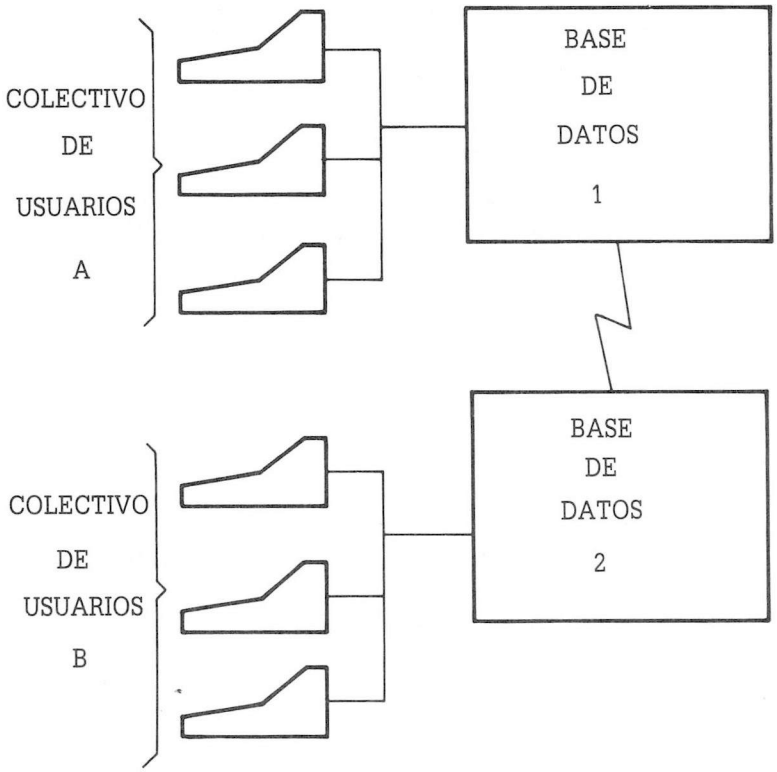


Figura 70. Mediante la unión de dos o más bases de datos se consigue formar un sistema de bases que permite a distintos colectivos utilizar conjuntamente toda la información del sistema.

La idea fundamental de las bases de datos estriba en la utilización conjunta de los mismos datos por el máximo número posible de aplicaciones y cuanto más amplio sea el número de datos y el número de aplicaciones coordinadas, más positiva será la explotación de la base de datos. La conclusión de este razonamiento es trivial: si desde el primer momento se diseña una base de datos capaz de almacenar todos los datos relacionados con el organismo tratado, el sistema será útil no sólo para las aplicaciones existentes o proyectadas, sino también para las que en el futuro se puedan necesitar. Este gran sistema debería estar almacenado en un lugar determinado e

interconectado mediante un sistema de telecomunicaciones con todos los posibles lugares de utilización.

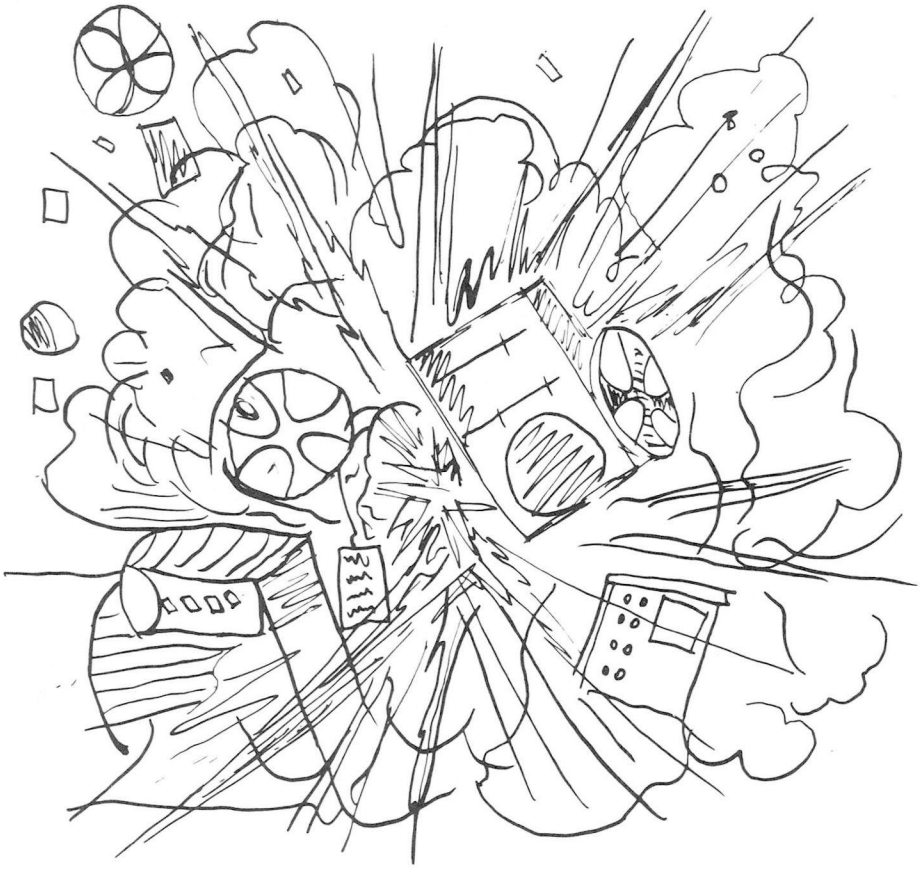


Figura 71. La creación de grandes sistemas de bases de datos (uniendo muchas bases de datos) es interesante desde el punto de vista teórico, pero debido al inmenso volumen de información que se tendría que manejar, se puede saturar las posibilidades reales de los ordenadores actuales.

En la práctica, la idea antes expuesta es inviable dado que exigiría un hardware y un software capaz de manejar un grandísimo número de datos, y en la actualidad no se dispone de tales equipos y programas. No obstante, los principales esfuerzos investigadores que se están haciendo en la actualidad, por los fabricantes de ordenadores y por los usuarios más potentes van encaminados a conseguir la integración total de datos, y tal vez en un futuro no muy leja-

no, la idea de estos macro-sistemas de bases de datos pueda ser implementable.

3.– UTILIZACION DE BASES DE DATOS

Probablemente una de las diferencias más importantes entre los ordenadores actuales y los de hace apenas una década de años, estriba en que los equipos modernos, deben estar preparados para la utilización masiva de datos.

El desarrollo de grandes sistemas de bases de datos, tanto a nivel experimental como a nivel práctico, está haciendo furor entre los usuarios no especializados en informática (también llamados usuarios finales). La importancia dada a la información, como recurso vital para la toma de decisiones, hace que se queden pequeñas las bases de datos particulares y por lo tanto las empresas, no solo utilizan sus propios sistemas, sino que también demandan información a otras bases de datos supra-nacionales.

El hecho de que el manejo de grandes sistemas de bases de datos mediante ordenadores tenga problemas técnicos, obliga a las empresas que necesitan dichas informaciones a recurrir a procedimientos manuales de ayuda. No obstante, en la actualidad existen muchos sistemas de bases de datos capaces de aportar informaciones de forma sencilla y barata. En resumen podemos distinguir dos tipos de utilizaciones: la de Sistemas ajenos y la utilización de Sistemas propios.

3.1 Utilización de sistemas ajenos.

El crecimiento del número de bits almacenados con capacidad de acceso directo en cualquier gran instalación ha aumentado logarítmicamente en los últimos años y es previsible que siga aumentando en el futuro. Esto ha implicado una disminución del costo por bit almacenado. Por otro lado están surgiendo nuevas tecnologias (sobre todo se está utilizando la óptica) que permiten, además de nuevas reducciones en el costo, aumentar la velocidad de acceso a los datos.

Estos dos condicionantes están haciendo prever que la posibilidad de acceder a Sistemas de bases de datos de carácter general, va a ser (ya lo es en algunos casos) mucho más rentable que el manejo de bases de datos gestionados por el propio usuario. Uno de los ejemplos más significativos de ese hecho es el sistema de bases de datos bibliográficos, por un coste auténticamente insignificante se puede estar conectado a los más importantes centros bibliográficos

de todo el mundo, con la ventaja adicional de que las consultas se pueden realizar en un lenguaje seminatural. Si se compara el costo de las consultas necesarias con el que resulta de la gestión y actualización del sistema de bases de datos bibliográficos de forma aislada, la diferencia es abismal.

3.2 Utilización de bases de datos propias.

En cambio, en otros casos, la naturaleza particular de los datos obliga a mantener una base de datos propia de la empresa que la utilizará. Evidentemente las exigencias, tanto en volumen de información almacenada, como en operaciones realizables por unidad de tiempo, es muy inferior cuando se trata de satisfacer a un único usuario que cuando se tiene que dar servicio a múltiples usuarios distintos.

La tarea de diseñar una base de datos propia, es cada día más complicada si se pretende que la base dé un rendimiento óptimo. Los programas de utilidad necesarios para realizar el mantenimiento de la base de datos deben ser aprovechados al máximo para que el cociente costo/servicio sea lo más bajo posible. En cualquier caso, la labor de diseño implica la participación de un experto en el tema, puesto que las estructuras con que se pueden almacenar los datos son muy distintas y todas ofrecen ventajas e inconvenientes. El diseñador, al final, no tiene mas remedio que elegir una de las posibles opciones y para ello deberá realizar unas comparaciones basadas en su experiencia.

En la actualidad, es frecuente encontrar bases de datos empresariales que no satisfagan plenamente los objetivos que se marcaron en su nacimiento, y ello suele ser consecuencia de un diseño erróneo.

3.3 El administrador de la base de datos.

En el anterior apartado describíamos los problemas existentes a la hora de diseñar una base de datos. Ahora nos vamos a encargar de los problemas existentes cuando la base ya ha sido diseñada e implantada.

Aunque se trate de una base de datos particular de una empresa, es de suponer que van a ser varios los usuarios de la misma que la utilicen e incluso varias las aplicaciones por cada usuario. Para controlar y coordinar la estructura de todos los datos, surge una nueva figura: el administrador de datos.

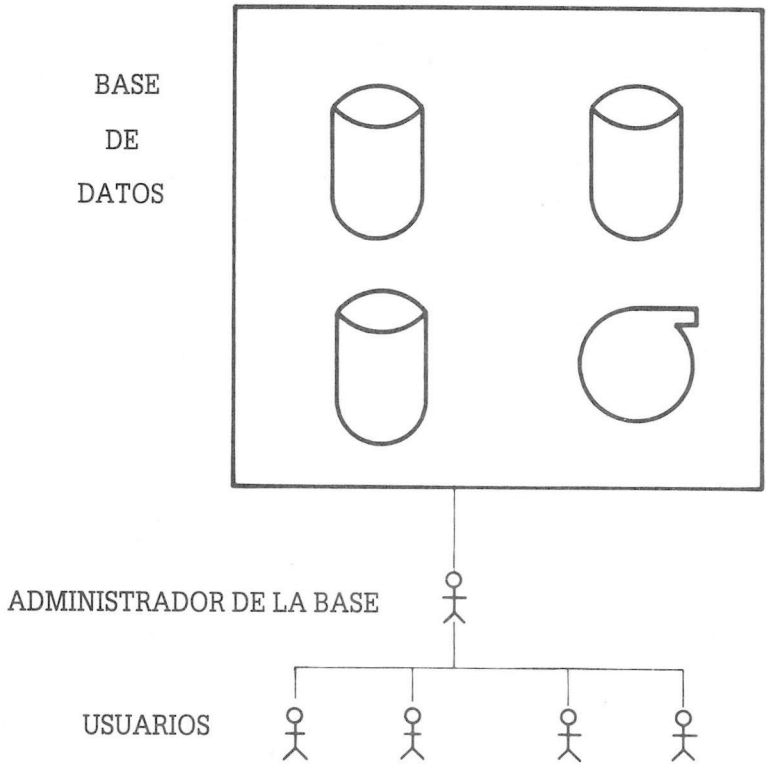


Figura 72. El administrador de la base de datos controla y coordina la estructura de la información. De alguna manera se puede afirmar que es un intermediario entre los datos de la base y los usuarios que los utilicen.

Es importante destacar que el administrador (puede ser un grupo de personas aunque nos referiremos a él en singular) no es el propietario de los datos, éstos pertenecen a los usuarios finales, sino el gestor que se encarga de facilitar la utilización comunitaria de los mismos. Para aclarar este concepto, podemos establecer una comparación entre el administrador de la base y el director de un banco. Evidentemente el director de un banco se encarga de gestionar los fondos depositados por sus clientes, para poder producir unos intereses, en algunos casos incluso aconseja sobre la forma óptima de invertir dichos fondos, pero en ningún caso se le puede considerar su dueño. Un defecto muy extendido entre las empresas que utilizan bases de datos es precisamente éste, el administrador se convierte en un tirano que, en vez de facilitar una utilización a gusto del usuario, impone sus criterios.

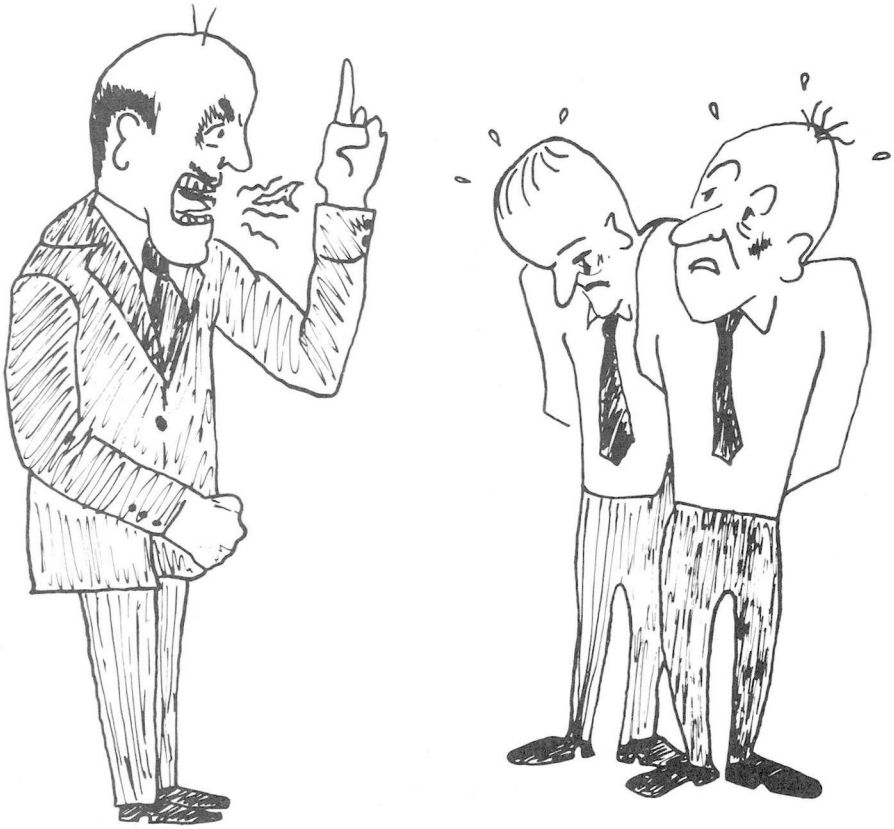


Figura 73. Es importante destacar que el administrador de la base de datos debe servir para ayudar al resto de usuarios. En algunas instalaciones, debido a la importancia de su cometido, el administrador se convierte en un tirano para los usuarios. Esta posibilidad debe ser evitada en todos los casos.

4.– REDUNDANCIA

Cuando dábamos una definición de base de datos citábamos tres condiciones necesarias para poder considerarla como tal. Además de esas tres condiciones incluiamos como condicionante principal el que los datos se encontraran almacenados en una única colección sin redundancias innecesarias.

Podemos tomar como definición de redundancia la repetición de datos en más de una ocasión. En prácticamente todos los bancos de datos que utilizan ficheros, hay una sorprendente calidad de datos duplicados o redundantes. En la mayoría de los casos el mismo dato

se encuentra almacenado en varios ficheros simultáneamente, según las distintas aplicaciones que lo utilicen. El problema no es solo físico (pérdida de capacidad de almacenamiento) sino también lógico (las fechas de actualización suelen ser distintas, por lo que una visión conjunta del mismo dato en los diveros ficheros produce resultados incongruentes).

Mediante la utilización de una base de datos se pretende eliminar esta redundancia o mejor dicho dejarla limitada a los casos en que sea necesaria. El hecho de que se utilice una base de datos está basado en el intento de obtener resultados rápidos y fiables y, precisamente por esto, en algunos casos merece la pena permitir una cierta redundancia si a cambio obtenemos mejores tiempos de acceso o mayor seguridad de que la información no será destruida accidental o intencionadamente. Por lo tanto, hay que admitir una cierta redundancia en las bases de datos, siempre que sea controlada y necesaria.

Los inconvenientes producidos por la redundancia en los sistemas tradicionales basados en ficheros son muchos y muy peligrosos:

1.– Para actualizar un dato redundante es necesario realizar múltiples operaciones de actualización (una por cada fichero en que se encuentre).

2.– Si, como suele ser habitual, las actualizaciones de los datos redundantes son realizadas por distintas aplicaciones, es probable que no se ejecuten todas simultáneamente (la diferencia normal suele ser de varios días, aunque hay veces que sobrepasa el mes) y por consiguiente los resultados obtenidos serán diferentes según la aplicación que se utilice. El usuario final tendrá un problema planteado ¿cual de los datos o más datos incongruentes es el fiable?. Probablemente su respuesta será, ninguno y perderá la fe en el sistema informático.

3.– Al tener la información almacenada varias veces se producirá un aumento del costo de almacenamiento.

En definitiva, la redundancia ha sido siempre uno de los enemigos más peligrosos de los sistemas informáticos. En las instaciones que comenzaban la mecanización partiendo de sistemas manuales, no existía redundancia en las primeras aplicaciones desarrolladas, pero a medida que aumentaban las actividades mecanizadas, aumentaba también la redundancia hasta llegar a niveles que hacia prácticamente imposible la explotación de las aplicaciones.

Aunque las bases de datos no han venido a resolver totalmente el problema de la redundancia, mediante su utilización es posible

controlarla y reducirla tanto como interese.

FICHERO DE PERSONAL

...	NOMBRE	SUELDO	PRECIO FACTURACION	...
...	**JUAN**	1.950.000	**1.900**	...
...	**LUIS**	2.500.000	**2.300**	...
	⋮	⋮	⋮	
...	PEDRO	1.500.000	1.350	...

FICHERO DE FACTURACION

...	CLIENTE	NOMBRE EMPLEADO	PRECIO FACTURACION	...
...	BANCO X	**JUAN**	**1.900**	...
...	EMPRESA A	**LUIS**	**2.300**	...
	⋮	⋮	⋮	
...	EMPRESA Z	**JUAN**	**1.900**	...

Figura 74. La redundancia, además de producir una sobrecarga inútil en el espacio de almacenamiento, puede conducir a errores si la actualización no se realiza simultáneamente en todos los ficheros. En la figura puede apreciarse en negrita los datos redundantes de dos ficheros independientes.

5.– MODIFICACIONES EN LA ORGANIZACION

Otro de los grandes problemas de las aplicaciones que se basaban en ficheros tradicionales era la realización de modificaciones en los mismos. Cada vez que había que añadir nuevos datos a un fichero era imprescindible modificar todos los programas encargados de actualizarse. Esto que a simple vista no parece un problema demasiado grande, si lo es, y para cerciorarnos vale con apuntar que el vo-

lumen de horas dedicadas a realizar modificaciones sobre aplicaciones ya existentes, por necesidades de ampliación de ficheros, alcanza el cuarenta por ciento de todas las horas disponibles en una empresa media con un nivel de mecanización también medio.

Los analistas y programadores que se encargaban de aplicaciones con ficheros, solían tomar como medida precautoria el añadir en cada registro del fichero un espacio reservado para futuras ampliaciones. Gracias a ello, en algunos casos a la hora de ampliar el fichero, las modificaciones a realizar en los programas eran menos drásticas, aunque en cualquier caso era necesario retocar todos los programas. Además, si la ampliación superaba el espacio reservado la preocupación inicial no producía prácticamente ningún alivio en las modificaciones a efectuar en las aplicaciones.

ESTRUCTURA INICIAL DE LOS DATOS DE PERSONAL

	NOMBRE	SEXO	SUELDO	
....	JUAN	VARON	2.500.000	.....
....	LUIS	VARON	3.000.000	.....
....	MARIA	HEMBRA	1.500.000	.....
	⋮	⋮	⋮	
....	PEDRO	VARON	1.900.000	.....

⇓

ESTRUCTURA DE LOS DATOS DE PERSONAL MODIFICADA

	NOMBRE	SEXO	EDAD	SUELDO	
....	JUAN	VARON	35	2.500.000	....
....	LUIS	VARON	49	3.000.000	....
....	MARIA	HEMBRA	28	1.500.000	....
	⋮	⋮	⋮	⋮	
....	PEDRO	VARON	16	1.900.000	....

Figura 75. Una de las principales ventajas incorporadas con las bases de datos es, que modificaciones en la estructura de la información no implican modificaciones en todas las aplicaciones que usen los datos. En el ejemplo de la figura, la modificación introducida consiste en la inclusión de un nuevo dato para cada empleado (la edad).

Otra de las características más importantes de las bases de datos es precisamente que soportan fácilmente las modificaciones o ampliaciones de la estructura de los datos contenidos, sin que sea necesario remodelar todos los programas. Esta ventaja no sólo es

importante por la comodidad que representa el no tener que escribir nuevos programas cada vez que uno de los usuarios de la base decida ampliar o modificar su estructura, sino que también deja tiempo libre en los programadores para que puedan dedicarse a desarrollar nuevas aplicaciones en vez de retocar otras ya existentes.

A esta característica de facilidad para las modificaciones también se le llama independencia entre datos y programas, ya que mediante determinados mecanismos, que veremos más adelante, es posible modificar unos y otros de forma aislada.

6.– NECESIDADES DE SEGURIDAD

Hasta ahora, la utilización de las bases de datos sólo nos ha reportado ventajas, ya que hemos eliminado problemas existentes con los tradicionales bancos de datos. En este punto vamos a estudiar un caso contrario. Al disponer de muchos datos interconectados y utilizados por varios usuarios, la seguridad de los datos es mucho más compleja de garantizar que cuando se trataba de un único fichero utilizado por un único usuario.

Figura 76. Las necesidades de seguridad en una base de datos, deben responder tanto a los casos en que un usuario inexperto pueda producir errores por accidente, como cuando elementos extraños a la organización pretendan hacer una mala utilización de la información.

Los niveles de seguridad dentro de las bases de datos tienen que atender a dos aspectos distintos. Por un lado deben asegurar la confidencialidad de algunos de sus datos, y en consecuencia, no permitir su visualización más que a los usuarios autorizados y por otro lado, se tiene que dar la posibilidad de recuperar la información en caso de pérdida por accidente.

La recuperación de datos no sólo debe ceñirse al dato propiamente dicho, sino que debe garantizar también que se respeten las múltiples interconexiones entre los datos, de esta forma se permitirá tanto la reutilización por el usuario propietario del dato, como por el resto de los usuarios autorizados.

7.– INDEPENDENCIA DE DATOS

La base de datos de una empresa no se puede considerar como una colección de datos estáticos. La manera en que los datos se almacenan varía continuamente y sería inútil pretender que la empresa se tenga que organizar en función de la organización de la base de datos, al contrario, lo que es imprescindible es que la base de datos sea capaz de adecuarse a la organización cambiante de la empresa. Como lo que se pretende es que estos cambios en la base de datos no obliguen a modificar los programas ya existentes, es imprescindible que la información esté almacenada a nivel de elemento. Así se puede conseguir que los registros lógicos que utiliza un programa sean distintos a los que utiliza otro, en definitiva la independencia entre los datos y los programas estriba en la posibilidad de que cada programa defina sus registros lógicos particulares independientes de la forma en que están los elementos dentro de la base de datos.

Cuando un programador comienza a realizar los programas de una aplicación, su visión de la base de datos es simple y muy parecida a la que tenía con los ficheros tradicionales. El tiene a su disposición los ficheros lógicos (se componen de registros lógicos) que necesita. Esta condición implica que en la base, los mismos datos pueden derivarse en distintos ficheros lógicos y por lo tanto, las estructuras del almacenamiento físico son muy complejas.

En el párrafo anterior, hemos llegado a dos conclusiones aparentemente contradictorias; para el programador su visión de la base de datos es simple, en cambio la organización entructural de la base es muy compleja. Evidentemente debe haber "algo" que permita al programador desinteresarse de esa complejidad y ese "algo", es el software de bases de datos. Estos programas, que son los que gestionan la base de datos, deben ser capaces de realizar las

transformaciones necesarias sobre la estructura de la base para que el programador pueda disponer del fichero lógico deseado. Existen dos políticas en los centros de cálculo donde se trabaja con bases de datos, una consiste en prohibir a los programadores que conozcan los detalles físicos del almacenamiento en la base; la otra obliga a los programadores a interesarse por la auténtica estructura física de la base. Como en casi todo, en el término medio está la virtud. Lo ideal es que el programador tenga un cierto conocimiento de los detalles físicos con objeto de evitar que se desaprovechen recursos, pero sin que este conocimiento sea tan exhaustivo que el programador intente programar sin utilizar las ventajas que le ofrece el software de base.

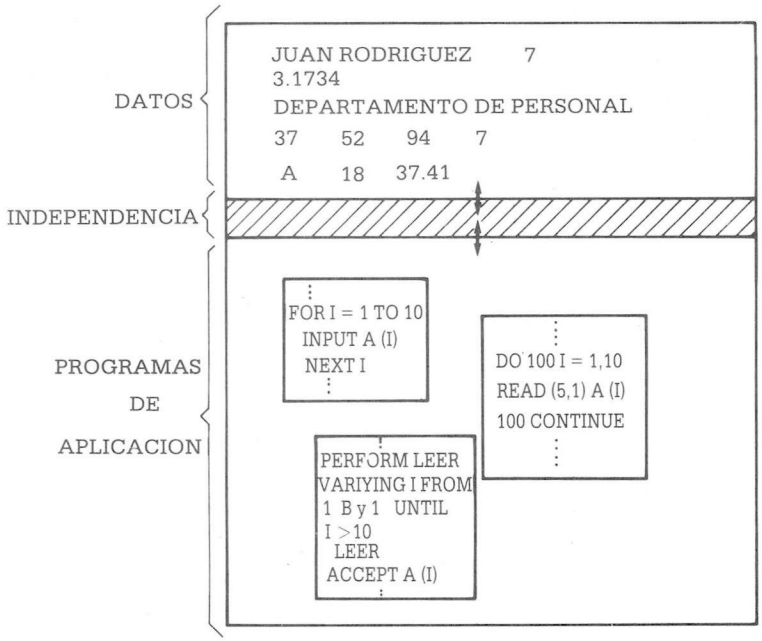


Figura 77. Mediante la utilización de bases de datos se consigue la independencia entre programas y datos. Gracias a esta independencia, la complejidad del almacenamiento físico puede ser ignorada por los usuarios finales.

De lo expuesto hasta ahora, se deduce que existen dos tipos de independencia entre datos y programas:

Independencia lógica:

Se dice que hay independencia lógica entre datos y programas cuando la modificación de la estructura lógica general de los datos, no afecta a los programas de aplicación.

Independencia física:

Se dice que hay independencia física entre datos y programas cuando la modificación de la distribución y organización física de los datos no afecta ni a la estructura lógica de la base ni a los programas de aplicación.

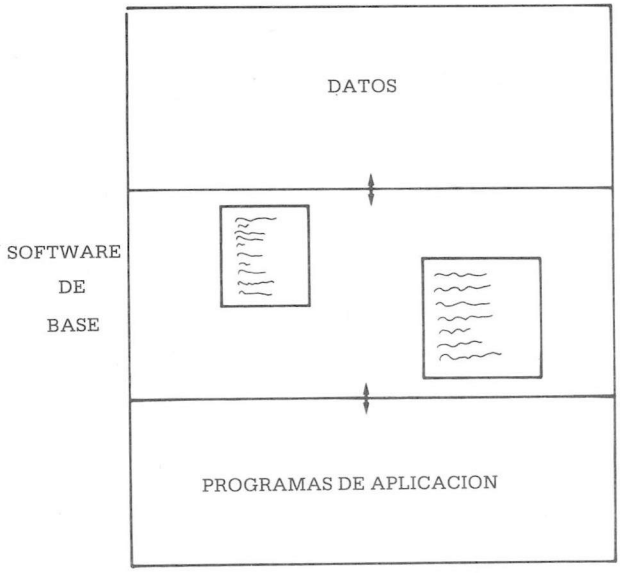


Figura 78. Para conseguir la independencia entre los datos y los programas de aplicación, es necesaria la intervención de otros programas específicos de manejo de la base de datos a los que se denomina software de base.

8.– PRINCIPALES VENTAJAS DE LA UTILIZACION DE BASES DE DATOS

El origen de las bases de datos fué debido a la necesidad de superar diversos problemas de los ficheros tradicionales. Para terminar este capítulo, dedicado a la introducción a las bases de datos, vamos a resumir cuáles han sido las principales ventajas aportadas:

Coste:

Evidentemente, la utilización de un sistema de bases de datos ofrece unas necesidades de almacenamiento muy inferiores a las necesarias según los sistemas tradicionales de ficheros. Esto se consigue gracias a la eliminación de datos repetidos y a que la organización física de los datos está realizada por el software de bases de datos con el objeto específico de minimizar las necesidades de almacenamiento.

Como consecuencia directa de esta minimización, resulta un coste sensiblemente inferior al que surgiría si se utilizasen ficheros independientes. Pero si tuviésemos que destacar el factor más importante a la hora de reducir costes, sin duda citaríamos las facilidades que ofrecen las bases de datos a la hora de programar nuevas aplicaciones. En la actualidad, el coste por unidad de información almacenada está bajando espectacularmente; en cambio, el coste de las horas necesarias para programación sube constantemente debido, entre otras cosas, a la inflacción. Por lo tanto son prácticamente más importantes las rebajas indirectas producidas sobre la programación que el ahorro producido directamente en el almacenamiento.

Redundancia:

El nivel de redundancia, antes de la aparición de las bases de datos, era altísimo en las aplicaciones informáticas de las grandes y medianas empresas. Como ya citamos antes, el objetivo de las bases de datos no es eliminar totalmente la redundancia, sino dejarla limitada a niveles bajos y controlados en cualquier caso. La ventaja producida por las bases de datos, en este aspecto, es doble. Por un lado reducen notablemente las necesidades de memoria auxiliar para almacenamiento y por otro, evitan los errores lógicos producidos al utilizar datos redundantes no actualizados simultáneamente.

Si bien en las bases de datos independientes se consiguen grandes descensos en la redundancia, cuando se unifican varias bases de datos para formar un sistema integrado la redundancia vuelve a aumentar. Este pequeño inconveniente es compensado por la inmensa capacidad de información que se obtiene con estas gigantescas organizaciones de datos.

Integridad:

En cualquier instalación surgen averías en el hardware que pueden hacer peligrar la información almacenada. Cuando una aplica-

ción trabaja con ficheros individuales, la recuperación de su contenido se puede realizar fácilmente si su dueño tuvo la precaución de realizar copias de seguridad. En cambio, las bases de datos contienen información de distintas aplicaciones y, además de guardar datos, guardan también las relaciones existentes entre los distintos elementos. Toda instalación con bases de datos debe garantizar la integridad del sistema, es decir, la recuperación de la información deteriorada por avería u otra causa, de forma sencilla y sin que sea necesaria la participación de los usuarios finales.

Al sistema de seguridad en las bases de datos no sólo se le exige que actúe en los momentos críticos, también se encarga de realizar controles periódicos en los que valida los datos almacenados, averiguando si existen incongruencias. Para ésto es imprescindible que de antemano se definan unas reglas sobre la relación existente entre los datos.

Independencia:

El volumen de datos contenidos en una base o sistema de bases es tan grande, que si se cuantifica en pesetas se necesitaría más de seis dígitos para expresar su valor. Por otro lado, la tecnología avanza a pasos agigantados y constantemente surgen nuevos equipos y programas. Por lo tanto, por razones técnicas o políticas, es fácil que una instalación decida cambiar a otro sistema de bases de datos. Pues bien, las bases de datos garantizan mediante la independencia física que esta evolución se podrá llevar a cabo sin perder todo el dinero invertido en la base de datos original.

Esta facilidad para la migración de datos puede ser utilizada también sin que la instalación decida cambiar de sistema. Una base de datos puede tener repartida su información en diferentes soportes, como discos magnéticos, cintas magnéticas, etc. El coste originado por el almacenamiento de un dato será muy distinto según el tipo de soporte en que se encuentre. Por ejemplo, un dato grabado sobre cinta magnética resulta mucho más barato que otro grabado sobre disco magnético. Los datos, como otros muchos recursos humanos, se ponen de moda en determinados momentos y decaen en otros. Evidentemente cuando un dato es muy utilizado, debe ser almacenado en un soporte rápido (por ejemplo, disco magnético), aunque resulte caro, en cambio cuando su utilización decae, su soporte ideal será secuencial (por ejemplo, cinta magnética). Las bases de datos facilitan la realización de migraciones cuando las necesidades de explotación lo determinen.

Rapidez:

Las bases de datos pueden utilizarse tanto para tratamientos en tiempo real, como para procesos por lotes. En el primer caso es fundamental que la rapidez de proceso sea alta para evitar que el usuario padezca tiempos de respuesta excesivos. En el segundo caso no hay usuario interactivo, pero en cualquier caso interesa que el caudal de transacciones sea elevado. Estos dos objetivos están correlacionados positivamente con la organización de los datos. Si la organización es buena, las búsquedas serán rápidas y en consecuencia, los tiempos de respuesta también serán rápidos. En cambio, una organización mala obligará a perder mucho tiempo en operaciones de búsqueda de datos y los tiempos de respuesta aumentarán.

El principal problema del diseñador para determinar la organización de la base estriba en el desconocimiento sobre las operaciones que necesitarán las futuras aplicaciones que trabajen sobre la base. Por lo tanto es necesario que el sistema sea lo suficientemente flexible como para permitir modificaciones en la organización de los datos, si la evolución del sistema base-aplicaciones así lo recomienda.

Para poder realizar estas modificaciones en la organización es imprescindible la utilización de programas que revisen automáticamente el funcionamiento de la base. En un futuro próximo estos programas vendrán incluidos dentro del software de base de forma que las reorganizaciones se realizarán automáticamente cuando el sistema lo estime oportuno.

Versatilidad:

Una de las características más apreciadas de las bases de datos, es la facilidad con que un programador puede definir y utilizar cualquier fichero lógico que necesite, independientemente de la estructura que tengan los datos implicados dentro de la base.

Esta característica viene derivada del método de organización que es el que debe garantizar la versatilidad del sistema. En algunos casos para permitir que la versatilidad sea grande, la organización debe ser extraordinariamente compleja.

Seguridad:

La integridad garantizaba la recuperación de los datos que pudieran haberse deteriorado por averías en el hardware. El concepto seguridad, en este caso, hace referencia a la privacidad de los datos. En algunos casos la información almacenada tiene gran valor, hasta

tal punto que es susceptible de robo o espionaje. Cuando más importante sea la información, mayor deberá ser el nivel de seguridad. Las dificultades encontradas en esta tarea son muy grandes ya que no sólo hay que luchar contra intrusos en la organización, sino que los propios integrantes de la empresa, en algunos casos, pueden intentar hacer un uso ilegítimo. Los niveles de seguridad pueden llegar a ser tan elevados que ni el propio administrador pueda visualizar los valores contenidos en la base.

En definitiva, la seguridad en una base de datos se refiere a la protección contra el acceso voluntario o involuntario por parte de usuarios no autorizados.

No obstante, algunos datos afectados por los niveles de seguridad pueden ser marcados como accesibles para determinados usuarios, es decir, el dueño de un dato puede determinar por sí mismo cuándo, cómo, y en qué medida se permitirá a otros usuarios visualizar o actualizar sus propios datos.

La seguridad es un tema extraordinariamente complejo que exige la colaboración del hardware y del software.

Simplicidad:

Aunque parezca un contrasentido, las aplicaciones que se apoyan en bases de datos, encuentran mayor simplicidad en éstas que en los ficheros individualizados. Gracias a los métodos utilizados para permitir que cada usuario pueda obtener visiones particularizadas de la base de datos, la enorme complejidad estructural de la base es inapreciable para los usuarios.

Una de las mejoras que ofrecen los sistemas de bases de datos es la facilidad para la búsqueda de datos. Con los ficheros tradicionales, las búsquedas dependían del tipo de fichero: en los secuenciales debían ser estrictamente sucesivas, en los de acceso directo se podía acceder por la clave de registro (clave principal) o por alguna de sus claves auxiliares. En los sistemas de bases de datos, se ofrece a los usuarios la posibilidad de realizar búsquedas por prácticamente cualquier dato, aunque esta facilidad resulta bastante costosa. Confiamos en que en el futuro se consigan equipos que posibiliten mayor rapidez en la búsqueda de la información.

CUESTIONARIO CAPITULO 5

CONCEPTOS FUNDAMENTALES SOBRE BASES DE DATOS

1.– ¿Por qué son necesarias las bases de datos?
2.– ¿Qué problemática presenta la independencia de ficheros?
3.– ¿Cómo se define un sistema de base de datos?
4.– ¿Cuál es el papel del Administrador?
5.– ¿Qué se entiende por redundancia?
6.– ¿Qué aspectos deben atender los niveles de seguridad?
7.– ¿En qué se diferencia la independencia lógica de la física?
8.– ¿Cuándo aumenta la redundancia?
9.– ¿Qué se entiende por integridad?
10.– ¿Qué significa independencia?

CAPITULO VI

ORGANIZACION DE LA INFORMACION

1.– INFORMACION

La información consiste en ideas y hechos acerca de diferentes cosas: personas, objetos, lugares, etc., y puede estar referida a tres mundos bien diferenciados que conviene estudiar por separado:

1.– El mundo real:
En él están contenidas las cosas sobre las que se tiene información.

2.– El mundo de las ideas:
Consiste en el conocimiento de las cosas del mundo real que existe en el cerebro de las personas, o en los ordenadores (por algo hace años se les denominaba cerebros electrónicos).

3.– El mundo de los datos:
En él se estructuran los datos que componen las ideas y se almacenan convenientemente.

Existe una terminología informática para hacer referencia a cada uno de estos tres mundos de información. A las cosas sobre las que se almacena información se les denomina entidades y pueden ser cosas tan tangibles como una mesa o un perro, pero también pueden ser tan intangibles como una tarea, la organización de un departamento de una empresa o cualquier otro concepto abstracto. Se puede afirmar que la información es lo que se piensa o se sabe acerca de las entidades. A menudo no interesa tener conocimiento de una úni-

ca entidad, sino que los conocimientos se extienden sobre diversas entidades homogéneas. Por ejemplo, continuando con la organización de un departamento de una empresa, lo usual es tener información sobre todos los departamentos de la empresa en cuestión; para ello se agrupa en un registro toda la información de cada entidad y se unen todos los registros asociados a las entidades homogéneas estudiadas. A estas colecciones de entidades similares se les denomina *conjunto de entidades*.

ENTIDAD	CONTEXTO
PERSONA	[ ] ES UN EMPLEADO DE NUESTRA EMPRESA, CUYA EDAD ES [ ] . ASIGNADO AL [ ] CON UN SALARIO DE [ ] Y CUYO DOCUMENTO NACIONAL DE IDENTIDAD ES [ ]

ENTIDAD	DATOS
PERSONA	JUAN RODRIGUEZ 31 AÑOS DEPARTAMENTO COMERCIAL 1.950.000 51.393.262

Figura 79. La información está compuesta por el contexto y los datos. Para determinar completamente el almacenamiento de la información, al contexto y los datos es necesario añadir la forma en que se representan éstos últimos.

En el mundo de las ideas se necesitan unos elementos para describir la información acerca del mundo real. Estos elementos reciben el nombre de *atributos de las entidades*. Los registros de las entidades contienen los valores de estos atributos que definen el cónocimiento. Los atributos en sí mismos, no indican ninguna propiedad sobre el mundo real, pero al ser unidos al contexto en que se definen, producen una representación del hecho sobre el que informan.

Por último, en el mundo de los datos hay que definir el soporte de la información. Esta puede hallarse registrada en tarjetas, archivadores, papel, etc. Dentro del ámbito informático existen dos posibilidades para almacenar la información. Una consiste en utilizar los tradicionales ficheros, en cualquiera de sus posibles modalidades (secuenciales, de acceso directo, etc.), y otra consiste en utilizar bases de datos, con lo que se consiguen todas las ventajas descritas en el capítulo anterior. Dentro del mundo de los datos, además de elegir la forma física de almacenamiento, también se deben tomar decisiones sobre la estructura lógica que se les debe dar para facilitar su manejo.

En resumen podemos destacar que en los últimos decenios, la importancia de la información se ha visto incrementada notoriamente y parece claro que la única forma de gestionarla rápida y eficazmente es mediante la utilización del hardware apropiado (ordenadores de la cuarta generación) y del software que facilite su manejo (programas para el mantenimiento de bases de datos).

2.– INFORMACION Y FICHEROS TRADICIONALES

La forma más común de almacenar la información mediante la utilización de ordenadores, es a través de ficheros tradicionales a los que también se pueden denominar ficheros planos. En ellos se incluye un registro por cada entidad homogénea de la que se tenga información. Cada registro se divide en un número fijo de campos que se utilizan independientemente o agrupados para almacenar los valores de los atributos de cada entidad. Para aclarar todos estos conceptos vamos a utilizar un sencillo ejemplo:

Sea una empresa (mundo real) en la que se quiere almacenar para cada uno de sus empleados (entidades homogéneas) el número de orden dentro de la empresa, el nombre y apellidos, el salario anual y el Departamento al que pertenezcan (atributos), y para ello se decide utilizar el siguiente fichero plano:

CAMPO

REGISTRO

13	LOPEZ GARCIA, P.	63	PLG
25	NIETO REY, C.	25	CNR
43	OTERO BLANCO, M.	53	MOB
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
91	PECES GAY, J.	42	JPG
127	RUIZ GARCIA, L.	81	LRG

FICHERO

Figura 80. Un fichero tradicional (también denominado fichero plano), consiste en una colección de registros sobre entidades homogéneas formados por un determinado número de campos en los que se almacena la información.

Cada fila del cuadro anterior se refiere a una entidad y forma un registro del fichero plano. Cada uno de los datos de una fila corresponde a un atributo y puede estar formado por uno o más campos.

Por lo tanto, los ficheros planos pueden compararse con matrices en las que cada fila corresponde a una entidad y cada columna a un atributo.

NUMERO DE ORDEN	APELLIDOS Y NOMBRE	SALARIO	DEPARTAMENTO
1	NIETO BALLESTEROS, SEGUNDO	1.900.000	INGENIERIA
2	JIMENEZ MARTIN, ENRIQUE	2.500.000	PERSONAL
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
67	GARCIA LOPEZ, JULIAN	1.850.000	INGENIERIA

Figura 81.

Con este tipo de ficheros tenemos perfectamente definido el mundo de los datos, ya que sabemos todos los valores de cada atri-

buto, pero desconocemos el contexto al que se refieren, pues en el fichero no se incluyen ni los nombres de los atributos, ni los tipos de atributos, ni la relación existente entre ellos. Por lo tanto, para obtener la información correspondiente habrá que complementar el fichero con las listas de estos datos no incluidos. Esta labor de complementar datos y contexto será responsabiidad de los programas que utilicen el fichero y de los usuarios que interpreten los resultados de los programas.

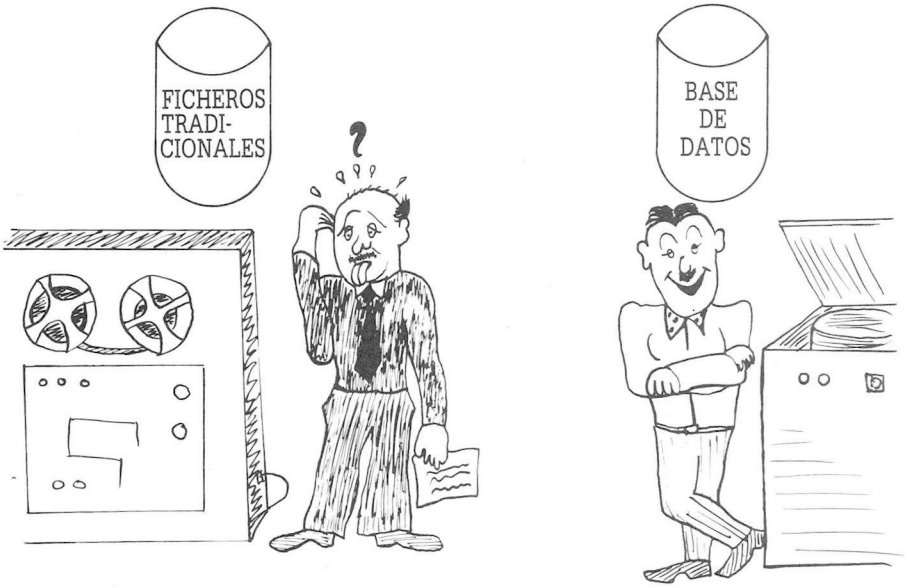


Figura 82. Cuando una instalación media llegaba a necesitar un número relativamente grande de ficheros, la redundancia crecía de tal forma, que dificultaba enormemente el mantenimiento. En estos casos, las bases de datos han venido a suponer una mejor utilización de la memoria de masa.

En definitiva, los ficheros tradicionales sólo sirven para almacenar datos, pero como ya vimos en el primer apartado de este capítulo, la información es mucho más que datos. En consecuencia, estos ficheros no son útiles como soporte de información, aunque esta afirmación no se puede considerar tajantemente. Si bien es cierto que la utilización de ficheros planos no es la más adecuada para almacenar grandes volúmenes de información, durante muchos años y aún en la actualidad, han servido para la gestión mecanizada de datos y su utilización sigue siendo indicada para el manejo de pequeñas cantidades de datos.

3.– INFORMACION Y BASES DE DATOS

Cuando se utiliza una base de datos para almacenar información, dentro de ella se incluirán no sólo los datos, sino también las distintas relaciones existentes entre ellas y el contexto en que pueden ser aplicados.

El concepto similar al de registro dentro de la terminología de bases de datos, es la *tupla*. Se llama tupla al conjunto de valores relacionados de alguna manera. Cualquier programa de aplicación o el administrador de la base de datos, puede referirse a una tupla o registro asociado a una entidad. Para ello es fundamental que el ordenador sea capaz de distinguir entre todas las tuplas almacenadas de entidades homogéneas. Para permitir esta localización cada tupla debe poseer una clave de identificación que permita distinguirla entre todas las demás. Al atributo clave de tupla se le suele denominar *identificador de entidad* y en algunos casos es posible que una entidad tenga más de un identificador.

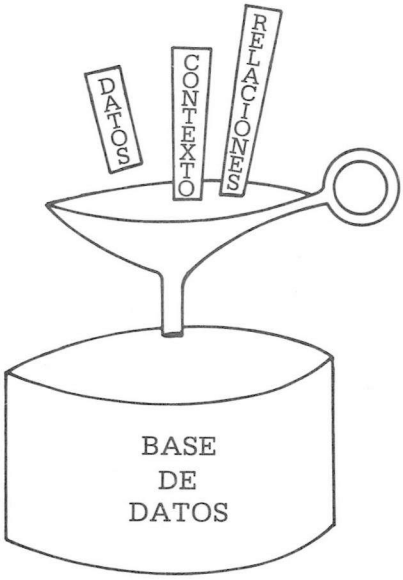


Figura 83. Aunque parezca un contrasentido, en las bases de datos no sólo se almacenan datos, sino que también se guarda el contexto en que pueden ser utilizados y las relaciones existentes entre ellos. De esta forma, lo que contiene la base de datos es auténtica información.

Si continuamos con el ejemplo anterior, la entidad "empleada" puede tener como identificador el atributo número de orden, ya que sin más que indicar un valor concreto de número de orden, cualquier programa será capaz de distinguir de forma inequívoca a una y sólo una de las entidades "empleadas". Sin embargo, el identificador de entidad en este caso no es único, puesto que el atributo "nombre y apellidos" también permite acceder al empleado deseado de forma inequívoca.

En resumen, se llama identificador de entidad al atributo o conjunto de atributos que el ordenador puede utilizar para identificar una tupla o registro.

Existen otros atributos dentro de la entidad que pueden ser utilizados para identificar a más de una tupla. A estos atributos se les denomina *identificadores auxiliares*. Un identificador auxiliar, dentro del fichero plano de personal, puede ser el departamento ya que al indicar al ordenador un departamento concreto, éste no es capaz de acceder a un único empleado, pero sí puede acceder a un grupo concreto de entidades, es decir, a todos los empleados que pertenezcan al departamento indicado.

A menudo, un archivo dispone de varios identificadores secundarios y mediante la utilización conjunta de ellos, se pueden imponer unas restricciones que sólo serán verificadas por un determinado subconjunto de tuplas. Esta posibilidad resulta especialmente útil cuando se desea localizar un colectivo de entidades homogéneas que satisfagan ciertos requisitos.

Para terminar con este apartado, podemos destacar que existen dos formas de "interrogar" a un conjunto de entidades homogéneas. La primera consiste en preguntar qué información se dispone de una entidad conocida; para ello vale con indicar el valor de un indentificador de entidad y automáticamente se obtendrá la tupla asociada y, por lo tanto, los valores de los atributos incluidos en la tupla. La segunda forma de consultar al conjunto de entidades es preguntando qué entidades verifican una propiedad dada. Esta segunda forma de interrogar puede resultar muy difícil para el ordenador, si en el diseño de la base de datos no se incluye un conjunto de entidades invertidas. Se denomina *conjunto de entidades invertidas* a un fichero en el que se incluyen todos los identificadores de entidad asociados a cada valor de atributo. Si en el diseño de la base se decide crear un conjunto completo de entidades invertidas, las búsquedas en la base con el segundo criterio serán muy sencillas, puesto que sin más que indicar el valor de cualquier atributo se pueden localizar en el

conjunto de entidades invertido todos los identificadores asociados, y posteriormente acceder según el primer criterio de interrogación a cada una de las tuplas implicadas. A cambio de esta sencillez, la creación de un conjunto completo de entidades invertido resulta muy caro, puesto que requiere gran cantidad de espacio para almacenamiento.

En la práctica, lo más usual es incluir en el diseño de la base de datos, conjuntos parciales de entidades invertidas en los que tan solo se tendrán en cuenta los valores de ciertos atributos. Con esta restricción se disminuye notoriamente el costo, pero se corre el riesgo de que en determinados momentos se desee realizar una consulta por algún atributo no considerado. No obstante, este problema se puede obviar ya que la definición de conjuntos invertidos es dinámica y puede ser realizada en cualquier momento.

Es importante destacar que un fichero invertido no es una base de datos, aunque las bases de datos sí utilizan ficheros invertidos.

FICHERO ORIGINAL:

NOMBRE	EDAD	DOMICILIO
LOPEZ	28	MADRID
PEREZ	29	TOLEDO
MORENO	28	TOLEDO
GARCIA	30	CUENCA
SANCHEZ	29	MADRID
RUIZ	28	MADRID
NIETO	30	TOLEDO
SANZ	30	CUENCA

FICHERO INVERTIDO POR EDADES

28	LOPEZ	MORENO	RUIZ
29	PEREZ	SANCHEZ	
30	GARCIA	NIETO	SANZ

FICHERO INVERTIDO POR DOMICILIO

MADRID	LOPEZ	SANCHEZ	RUIZ
TOLEDO	PEREZ	MORENO	NIETO
CUENCA	GARCIA	SANZ	

Figura 84. Los ficheros invertidos son una de las "herramientas" utilizadas por el software de bases de datos. Cuando se utilizan ficheros invertidos, independientemente de una base de datos, pueden resultar muy útiles a la hora de localizar una información determinada; no obstante, no deben confundirse con una auténtica base de datos.

En este apartado dedicado a la información y bases de datos, hemos evitado utilizar el término "fichero" y en su lugar hemos usado "conjunto". Este hecho está basado en que, como ya hemos visto anteriormente, en las bases de datos los ficheros lógicos pueden ser definidos libremente por el programador y, en el almacenamiento físico, los datos no se agrupan en fichero, sino que se almacenan los elementos individualmente.

4.– RELACIONES ENTRE LOS DATOS

La misión de las bases de datos, en cuanto al almacenamiento de información, se centra en guardar tanto los datos propiamente dichos, como las relaciones existentes entre ellos. Precisamente esta es una de las ventajas que surgen al utilizar bases de datos en vez de ficheros tradicionales. Esta desconexión entre el almacenamiento físico y ficheros lógicos de usuario, acarrea una cierta complejidad en cuanto a la representación lógica de los datos.

El usuario final puede utilizar cualquier representación lógica independiente de la base de datos. Lo único que debe indicar al ordenador son las características de su visión de la base. A partir de estas características, el software de base se encargará de establecer las conexiones necesarias entre la forma en que estén los datos almacenados y la forma en que desea utilizarlos el usuario.

Existen muchos métodos distintos para que el usuario defina ficheros lógicos y, aunque teóricamente éste no tiene ninguna limitación a la hora de decidir sus registros lógicos, en la práctica existen ciertas restricciones que vienen marcadas por el software básico de bases de datos. En cualquier caso, la organización lógica del usuario debe ser formalmente definida.

Esquema:

Se define esquema como la descripción lógica de la base de datos. En él deben especificarse los nombres de todas las entidades, los nombres de los atributos de cada entidad y las relaciones existentes entre ellos. En definitiva, mediante el esquema de una base de datos, se pretende dar un "mapa" de los datos, en el cual las "carreteras" serían las relaciones entre los datos, cada "provincia" sería una entidad y los atributos de cada entidad serían los "municipios" de cada provincia.

Cuando se define el esquema de una base de datos, no se debe hacer referencia a los valores que están almacenados o se puedan al-

macenar en el futuro en la base. Tan solo se pretende referenciar los nombres de los datos y sus relaciones. Cuando un programa haga referencia a este esquema y rellene los nombres de los atributos con datos concretos, se dice que tenemos una *instancia del esquema.*

La forma de representar el esquema de una base de datos es mediante registros lógicos en los que agrupar los atributos. Al igual que en el caso anterior, se dispone de un esquema de registros lógicos que no incluye ningún valor en concreto (sólo se compone de los nombres de los atributos del registro). Cuando un programa rellene estos atributos con valores concretos, se dice que tenemos una *instancia de registro lógico.*

Para aclarar estos conceptos y los que surgirán en próximos apartados, vamos a definir el esquema de la base de datos de una hipotética empresa dedicada a la compra/venta de productos. Debe quedar muy claro que esta definición está hecha únicamente con fines didácticos y en cualquier caso real, la complejidad del esquema sería muy superior.

Supongamos que nuestra empresa maneja sólo cinco registros distintos: ALBARANES DE VENTA, DETALLE DE PRODUCTOS, CLIENTES, PRECIO DE VENTA y PRODUCTOS.

En los ALBARANES DE VENTA se indica el número de orden de venta, el número del cliente, la fecha de entrega y el porcentaje de descuento.

Para cada albarán de venta, se deben especificar qué productos son los vendidos y ésto se hace mediante el DETALLE DE PRODUCTOS, que contiene como atributos el número del producto y la cantidad vendida.

En el registro CLIENTES, se incluyen los datos generales de los clientes de la empresa, como número de cliente, nombre y dirección.

Para cada cliente se indica los precios de los productos que tiene pendientes de servir mediante el registro de PRECIOS DE VENTA que dispone de los siguientes datos:

– Número del producto y precio de venta.

Por último en el registro de PRODUCTOS, se almacenan los datos de los productos manejados por la empresa: número del producto, nombre del producto y stock existente.

Según las anteriores especificaciones, el esquema de la base de datos sería el siguiente:

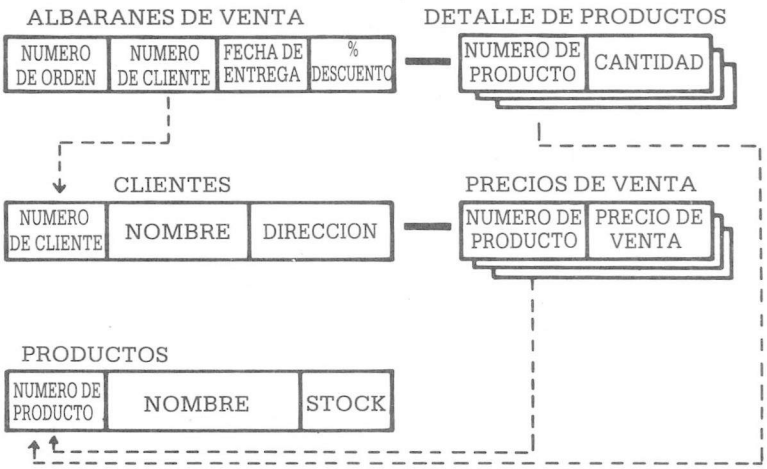


Figura 85. Los usuarios de la base de datos suelen tener conocimiento de la estructura lógica a través de los subesquemas, aunque, en algunos casos, también pueden obtener la representación completa ofrecida por el esquema.

Como se puede observar, para representar el esquema se ha elegido un diagrama de bloques, asociando un bloque a cada tipo de registro e incluyendo en él los atributos correspondientes. Cuando dos bloques están relacionados directamente, se unen mediante una línea continua. Por ejemplo el registro ALBARANES DE VENTA está relacionado con registros DETALLE DE PRODUCTOS, que son los que especifican los productos vendidos en cada albarán. El registro CLIENTE está relacionado con registros PRECIOS DE VENTA que informan de los precios de venta de los productos para cada cliente.

Cuando existen referencias cruzadas entre dos registros, es decir, cuando a través de un atributo de un registro se puede localizar otro registro de otro tipo asociado a él, estas se marcan mediante una línea discontinua. Por ejemplo, en el registro ALBARANES DE VENTA, no figura ni el nombre ni la dirección del cliente, pero mediante el atributo NUMERO CLIENTE se establece una referencia cruzada con el registro CLIENTE, donde sí figuran esas informaciones. Análogamente en los registros DETALLE DE PRODUCTOS y PRECIOS DE VENTA, no figura ni el nombre del producto ni el stock existente, pero mediante el atributo NUMERO DE PRODUCTO se puede establecer una referencia cruzada con el registro PRODUCTOS, donde sí se encuentra almacenada esa información.

Las líneas continuas, es decir, las relaciones entre distintos tipos de registros, deben estar obligatoriamente incluídas en el esquema, dado que sin ellas sería imposible completar la información, en cambio las líneas discontinuas que indican las referencias cruzadas, no agregan ningún conocimiento fundamental ya que sin más que repasar el contenido de todos los tipos de registros, se pueden establecer automáticamente las referencias cruzadas existentes. En definitiva la única utilidad de las líneas discontinuas es facilitar el camino de acceso a la información.

Subesquema.

La visión ofrecida por el esquema es total, es decir, proporciona un diagrama general de todos los tipos de registros de la base de datos. Sin embargo, la visión de la base de datos, bien porque no interese proporcionar toda la información por seguridad o bien por facilitar el trabajo de los programadores, puede ser parcial para algunos usuarios. En estos casos se utiliza un subesquema en vez de un esquema. Es decir, se define subesquema como una descripción lógica de parte de la base de datos. Evidentemente del mismo esquema se pueden deducir varios subesquemas y la incorporación de nuevos subesquemas se realizará de acuerdo con las necesidades de los usuarios, teniendo en cuenta que deben ser supervisados por el administrador de la base de datos.

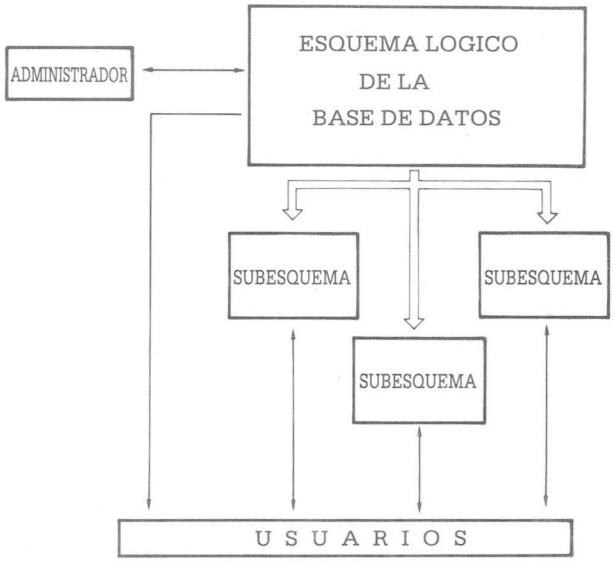


Figura 86.

A continuación vamos a definir dos subesquemas distintos pero ambos procedentes del esquema de base de datos utilizado en el apartado anterior.

Supongamos que un programador debe realizar un programa para la gestión de los albaranes de venta y que su visión deseada incluye sólo dos tipos de registro que decide denominar ALBARANES y DETALLE, en el primero desea incluir el número de orden del albarán, el nombre del cliente, la dirección del cliente y el porcentaje de descuento y en el segundo, quiere disponer del número de los productos pedidos, el nombre del producto, la cantidad pedida y las unidades existentes en stock. En este caso podría disponer del siguiente subesquema:

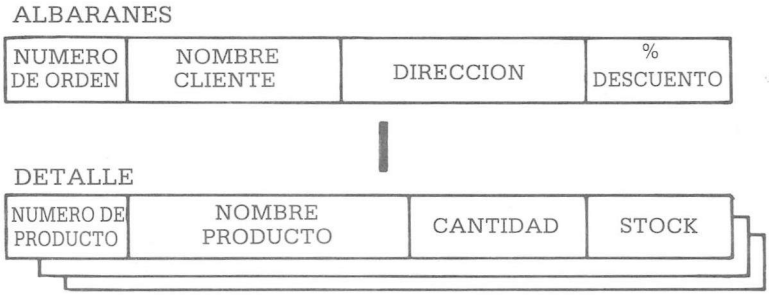


Figura 87.

Otro segundo programador, que debe realizar un programa para el control de facturación a clientes, tan solo desea disponer de dos tipos de registros, a uno le denomina MAESTRO DE CLIENTES y quiere que contenga el número de cliente y el nombre, y otro que denomina PEDIDOS, en el que quiere que figuren los números de los productos vendidos, su nombre y precio de venta. Para esta aplicación se puede definir un segundo subesquema de la siguiente manera.

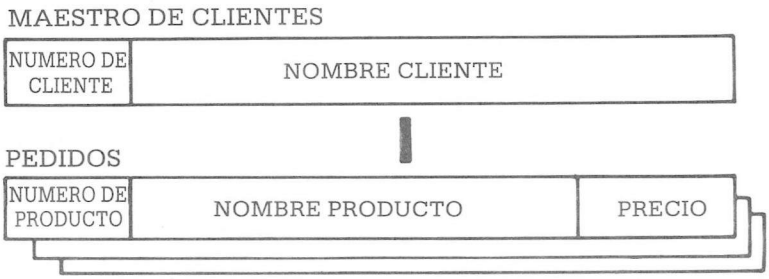


Figura 88.

Sin más que observar los dos subesquemas anteriores, se puede ver que la visión que dan de la base de datos es parcial y muy distinta una de otra. Precisamente éste era el objetivo perseguido, cada programador podrá desarrollar su programa sin tener en cuenta toda la complejidad de la base con lo que los programas quedarán muy reducidos y su programación será muy sencilla.

De alguna manera, aunque no sea rigurosamente cierto, se puede afirmar que el esquema de la base de datos es propiedad del administrador de la base, mientras que los distintos subesquemas derivados de él son propiedad de los usuarios finales.

5.– VISIONES DE LA BASE DE DATOS.

De todo lo expuesto en este capítulo, se deduce que existen diferentes visiones lógicas de la misma base de datos, según el usuario que la esté explotando. Estas visiones no tienen nada en común con la estructura física de los datos, que por considerarla excesivamente técnica no será tratada en este libro. Todas las visiones de la base se pueden agrupar en cuatro grandes apartados:

1.– Visión física de la base.

Representa la organización real de los datos dentro de los soportes de almacenamiento. Sólo es utilizada por los programadores de sistemas que se ocupan del mantenimiento de los sistemas operativos. En algunos casos puede ser conveniente que los diseñadores de la base de datos tengan acceso a esta visión con objeto de optimizar el rendimiento general de la base, que puede estar muy ligado a la organización física.

En resumen, esta visión permite examinar la organización del almacenamiento físico.

2.– Visión del administrador de la base de datos.

La forma de pasar desde la visión física a la visión del administrador es mediante las conversiones realizadas por el software básico. En ella se especifica la organización global lógica de la base de datos y, por supuesto, su utilización está restringida al administador de la base de datos.

Esta visión viene representada por el esquema de la base de datos.

3.– Visión de los programadores de aplicaciones.

Esta visión, al igual que la anterior, puede lograrse gracias al software básico de bases de datos. En ella se pueden realizar

particularizaciones que permitan a los programadores de aplicaciones obtener las mejores condiciones para implementar sus programas.

Las distintas visiones de la base que tengan los programadores de aplicaciones, vienen representadas en los subesquemas de la base de datos.

4.– Visión de los usuarios de terminal.

Este último punto de vista está reservado para el personal no informático que en definitiva utiliza la base de datos. Debe ser lo más parecida posible a la visión que el usuario tenía del trabajo desarrollado antes de la mecanización. La forma de llegar a esta visión es mediante los programas de aplicación y suele realizarse mediante terminales interactivos que permitan el diálogo hombre-ordenador.

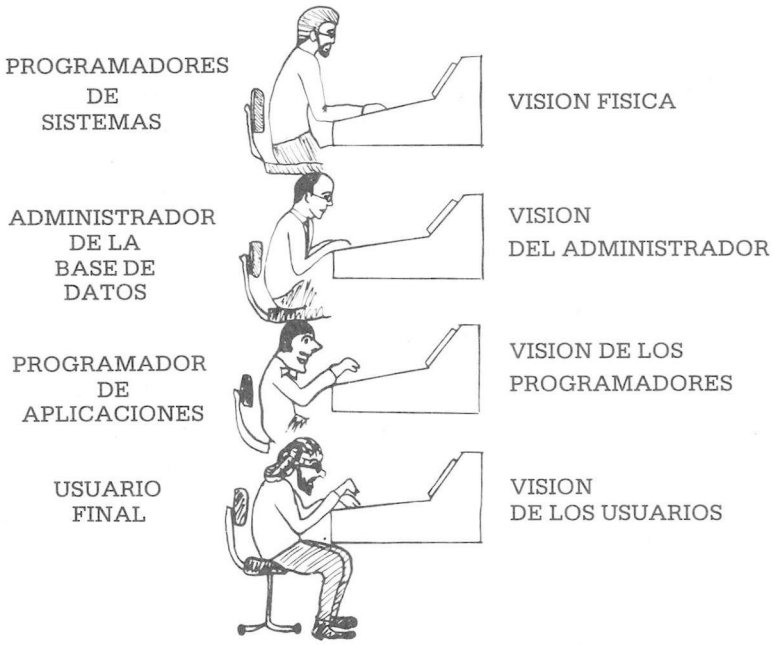


Figura 89. Las cuatro visiones lógicas de la base de datos dan la suficiente flexibilidad como para que cada especialista se encuentre cómodo trabajando, sin interferir unos en otros.

Las distintas visiones de los usuarios de terminal deben estar representadas en los manuales de usuario escritos por los

programadores de las aplicaciones.

Es importante destacar que cualquiera de estas cuatro visiones puede ser mofificada sin que ello afecte al resto. Mediante esta facilidad se evita el hecho que, por ejemplo, cuando un usuario final decida modificar sus aplicaciones, esta modificación afecte a la visión de otros usuarios.

6.– CORRESPONDENCIA ENTRE DATOS.

Entre dos tipos de datos puede haber tres correspondencias distintas:

1.– Correspondencia biunívoca:

Mediante este tipo de correspondencia, se establece una equivalencia uno a uno entre dos tipos de datos. Es decir, conocido un dato del primer tipo, inmediatamente podremos conocer el dato correspondiente del segundo tipo.

Un ejemplo claro de una correspondencia biunívoca está patente entre los siguientes tipos de datos: NUMERO DE CLIENTE y NOMBRE DE CLIENTE. Evidentemente si se conoce el número de un cliente dado, inmediatamente podemos averiguar su nombre y viceversa, a partir del nombre del cliente se puede averiguar su número.

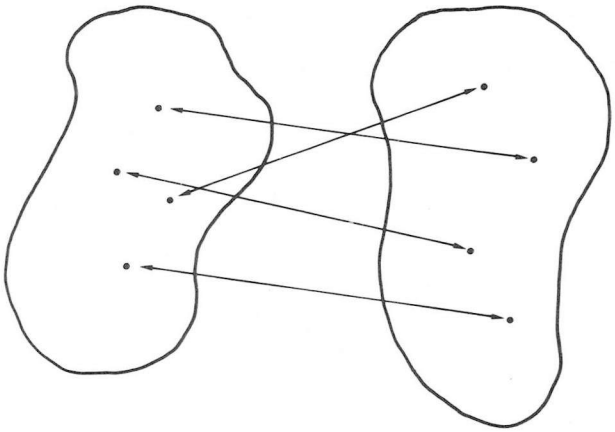


Figura 90. La correspondencia biunívoca permite establecer una relación entre dos conjuntos de atributos de tal forma que, conocido un elemento de cualquiera de los dos conjuntos, se puede determinar uno, y sólo uno, de los elementos del otro conjunto. A esta correspondencia también se le llama "uno a uno".

La forma más usual de notar esta correspondencia es interponiendo una flecha de dos direcciones entre los tipos de datos relacionados. Por ejemplo:

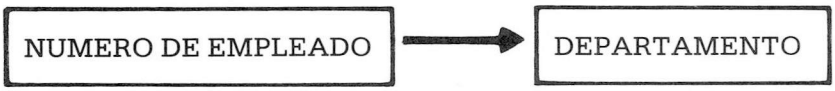
Cuando se da esta correspondencia, las características funcionales de ambos tipos son idénticas, por ejemplo si uno de ellos es identificador de entidad también lo será el otro.

2.– Correspondencia simple.

Mediante este tipo de correspondencia se establece una relación "no correspondencia", es decir, conocido un elemento de uno de los tipos de datos podemos conocer automáticamente el elemento correspondiente en el otro tipo de datos. Sin embargo, aún conociendo un elemento concreto del segundo tipo de datos no se puede identificar ningún elemento correspondiente en el primer tipo de datos.

Un ejemplo de esta correspondencia simple lo encontramos entre los tipos de datos NUMERO DE EMPLEADO y DEPARTAMENTO, ya que si conocemos el número de un empleado podemos, sin más que observar su registro, conocer el departamento al que pertenece, en cambio si conocemos un departamento concreto no podemos saber un número de empleado concreto.

La correspondencia simple se nota mediante una flecha que partiendo de tipo de datos "origen" llega al tipo de datos "destino". Por ejemplo:



La utilidad de esta correspondencia es evidente, permite un ahorro considerable de espacio de almacenamiento ya que todos los datos asociados al departamento, como podrían ser el nombre del departamento, su jefe, etc., no es necesario que se repitan en cada uno de los empleados. Vale con indicar para cada empleado el departamento a que pertenece y, en otra zona distinta se indicará una unica vez, la clave de departamento y sus datos asociados.

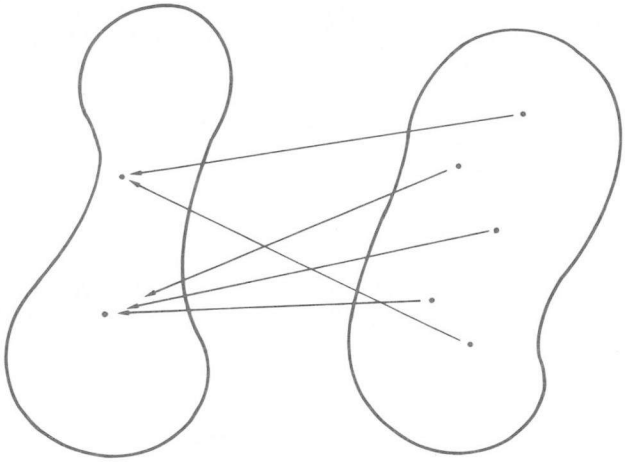


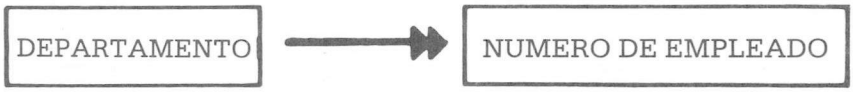
Figura 91. La correspondencia simple, ofrece la posibilidad de determinar el elemento correspondiente en el segundo conjunto a partir de un elemento del primero, pero no en sentido contrario. Otra denominación de esta correspondencia es "muchos a uno".

3.– Correspondencia compleja.

La correspondencia compleja es el caso contrario al de la correspondencia simple, conocido un elemento concreto de uno de los dos tipos de datos relacionados podemos conocer automáticamente varios elementos del otro tipo de datos.

Como ejemplo, para la correspondencia compleja, podemos tomar los mismos tipos de datos que en el caso anterior, es decir NUMERO DE EMPLEADO y DEPARTAMENTO, pero en este caso en sentido contrario, de forma que conocido un departamento concreto podemos hacerle corresponder de forma automática con varios números de empleado.

Para notar la correspondencia compleja de forma cómoda se utiliza la misma flecha que en la correspondencia simple pero con dos cabezas, asi se hace referencia a la existencia de más de un elemento correspondido. Por ejemplo:



La utilidad práctica de esta correspondencia se centra en la búsqueda de colectivos. Aunque también podría apuntarse la msima propiedad de ahorro de espacio que en el caso anterior.

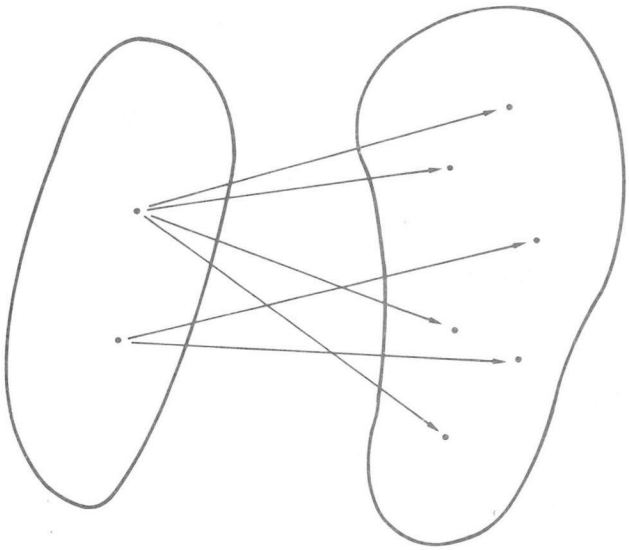


Figura 92. La correspondencia compleja relaciona a cada elemento del primer conjunto, con varios del segundo. Por eso también se le denomina correspondencia "unos a muchos".

CUESTIONARIO CAPITULO 6

ORGANIZACION DE LA INFORMACION

1.– ¿Qué es información?
2.– ¿Qué son los atributos?
3.– ¿A qué se llama fichero plano?
4.– ¿Qué es un identificador de entidad?
5.– ¿Qué es una tupla?
6.– ¿Los ficheros invertidos son bases de datos?
7.– ¿Qué es un esquema?
8.– ¿Quién utiliza los subesquemas?
9.– ¿Bajo qué puntos de vista puede contemplarse una base de datos?
10.– ¿Qué es una correspondencia biunívoca?

CAPITULO VII

MODELOS DE BASES DE DATOS

1.– MODELO JERARQUICO.

En el capítulo anterior introdujimos el concepto fichero plano, en él cada registro tenía una serie de campos de longitud fija e iguales en todos los registros. Precisamente por esta característica, el fichero se podía representar en una tabla de dos dimensiones y, de ahí le venía la denominación de fichero plano.

En muchas ocasiones resulta imposible conseguir que un fichero sea plano y para solucionar estos casos antes de la aparición de las bases de datos se recurría a dividir el fichero, que debía ser único por su naturaleza común, en varios ficheros independientes. Esta división contra-natura implicaba la aparición de problemas tanto en los procesos de actualización como en los de obtención de información.

Los sistemas de bases de datos han aportado varias soluciones a este problema. Cada una de las soluciones ha dado origen a un modelo de base de datos. En el primer apartado de este capítulo vamos a estudiar el modelo jerárquico que, como su propio nombre indica, se base en la jerarquización de los datos almacenados. Para realizar la jerarquización, el modelo utiliza un tipo de estructura bien estudiado por las Ciencas Matemáticas: el árbol.

1.1. Arbol

Un árbol consiste, en el mundo real, en una raiz y varias ramas, llamándose nudo a los puntos de unión entre las ramas. La similitud entre un árbol real y la estructura de datos denominada árbol es to-

tal. La estructura jerárquica entre los distintos nudos del árbol viene marcada por su proximidad a nudo raiz. De esta forma, la máxima jerarquía dentro del árbol (se suele representar invertido) está concentrada en el nudo raiz, el segundo nivel jerárquico será para los nudos descendientes directamente de la raiz, el tercer nivel para los descendientes de éstos y así sucesivamente.

Para centrar la situación de un nudo respecto a sus nudos vecinos se pueden determinar hasta cinco relaciones distintas:

1.– Padre.
   Se dice que un nudo A es padre de otro B, cuando B desciende directamente de A.
2.– Hijo.
   Se dice que un nudo A es hijo de otro nudo B, cuando A desciende directamente de B.
3.– Hermanos.
   Se dice que un nudo A es hermano de otros nudos B1,B2, ...Bn, cuando todos ellos desciende del mismo padre.
4.– Mellizos.
   Se dice que un nudo A es mellizo de otros nudos B1,B2, ...Bn, cuando todos ellos son hermanos y además son del mismo tipo.
5.– Hoja.
   Se dice que un nudo A es una hoja, cuando no tiene ningún hijo.

Puede parecer extraño que estemos mezclando terminología "familiar" y "botánica" para denominar a unos elementos homogéneos, pero esta terminología es la más extendida y no nos parece oportuno aportar nuevas definiciones más coherentes, pero que llevarían a la confusión al lector cuando estudie otras publicaciones.

Evidentemente dentro de un árbol normal, todo nudo no puede tener más que un único padre y sin embargo, todo nudo, puede tener varios hijos. El hecho de que los árboles se representen siempre en forma invertida, se basa en que de esta manera quedan por encima los nudos con mayor jerarquía y por debajo los nudos de menor jerarquía.

La utilidad de las estructuras arborescentes es especialmente importante para detallar las relaciones entre los distintos registros lógicos de la base de datos, aunque también pueden usarse para describir la organización física.

Para terminar de aclarar la definición de árbol, vamos a describir un sencillo ejemplo en el que marcaremos todos los tipos de nudos existentes.

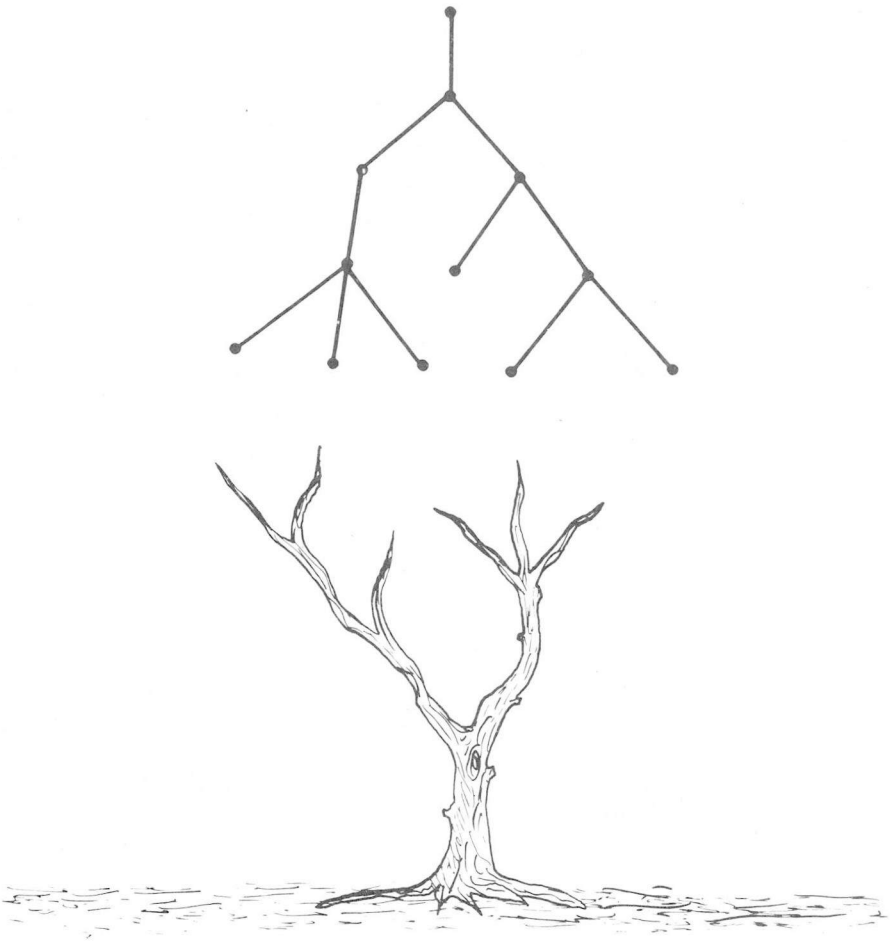


Figura 93. La similitud entre la estructura de datos denominada árbol y un árbol real, es grande. La única diferencia es que cuando se trata de un árbol de datos, se suele representar invertido para evidenciar la jerarquía entre los nudos.

Supongamos una empresa que tiene cuatro departamentos distintos: PERSONAL, ADMINISTRACION, PROMOCION e INGENIERIA. El departamento de personal consta de una única sección en la que están encuadrados tres empleados A,B y C; el departamento de administración tiene dos secciones, una CONTABILIDAD y otra

FACTURACION, en la primera están encuadrados dos empleados: D y E y en la segunda tres: F,G y H; el departamento de promoción tiene una única sección con dos empleados I y J; y por último el departamento de Ingeniería está formado por tres secciones, CIVIL, INDUSTRIAL e HIDRAULICA, la primera tiene cuatro empleados: K,L,M y N, la segunda cinco: O,P,Q,R y S, y la tercera dos: T y V. La organización del personal de dicha empresa se podría reflejar en el siguiente árbol:

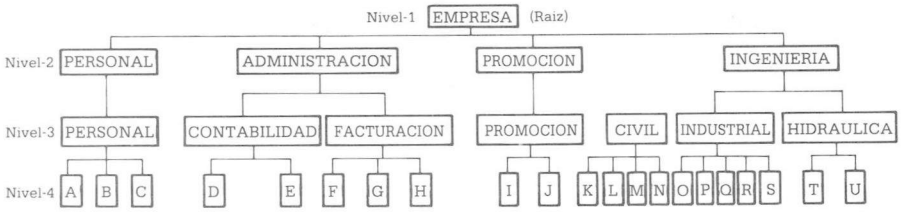


Figura 94.

En este árbol se puede ver que todos los nudos de empleados (desde A hasta U) son hojas y están en el nivel jerárquico inferior. La raiz, y por lo tanto nivel jerárquico superior, es el nudo de la empresa en general y, por ejemplo, el nudo INGENIERIA tiene tres hijos: CIVIL, INDUSTRIAL e HIDRAULICA, mientras que el padre de FACTURACION es ADMINISTRACION.

1.2 Tipos de árboles.

Dentro del conjunto de todos los árboles posibles, podemos establecer dos clasificaciones: una atendiendo a la disposición de las ramas y otra atendiendo a los registros de la base de datos asociados a cada nudo.

1.– Clasificación de árboles según la disposición de las ramas.

Se llaman árboles balanceados a aquellos en los que todos los nudos pueden tener el mismo número de ramas y los nudos se recorren de arriba a abajo y de izquierda a derecha. En caso contrario se dice que el árbol es desbalanceado.

Sería interesante, desde el punto de vista de la explotación, que los árboles que representan la organzación de una base de datos fueran balanceados, ya que al tener todos los nudos el mismo número de hijos, los algoritmos de búsqueda en el árbol son más sencillos y eficientes. Desgraciadamente, las

necesidades de organización exigen árboles desbalanceados por lo que los algoritmos deben ser más complejos.

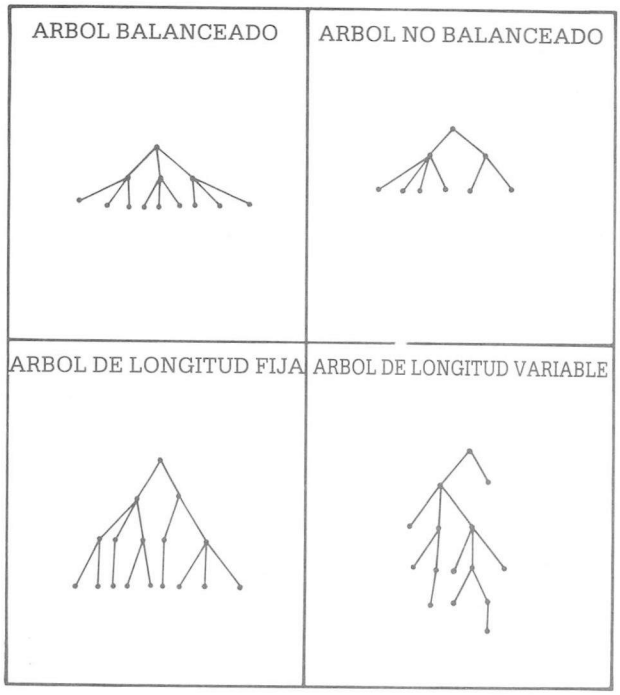


Figura 95. Dentro de las innumerables clasificaciones entre los árboles, podemos destacar los tipos de árboles de la figura. En los árboles balanceados cada nudo tiene un número fijo de descendentes. A los árboles que no verifican esta condición se les llama no balanceados. Por otro lado, un árbol es de longitud fija si todas las ramas pueden llegar a un determinado nivel, en cambio los de longitud variable no cumplen esta condición.

Dentro de los árboles balanceados, existen unos especialmente cómodos para su explotación, y que en ciertos casos son muy apropiados para representar organizaciones reales: los denominados árboles binarios. La propiedad que cumple este tipo de árboles es que cada nudo admite sólo dos ramas y por consiguiente dos descendientes.

Los árboles binarios encajan perfectamente en la organización física de los datos, ya que la naturaleza binaria del ordenador facilita el ajuste de determinados procesos con estos árboles.

2.– Clasificación atendiendo a los registros asociados a cada nudo.

Como ya decíamos anteriormente, las estructuras arborescentes se pueden utilizar para representar tanto la organización física como lógica de una base de datos. Para ello se puede establecer una equivalencia entre los nudos del árbol y los registros físicos o lógicos de la base.

Según cómo se realice esta equivalencia, podemos clasificar los árboles en homogéneos o heterogéneos. Los primeros deben cumplir la propiedad de que todos los nudos del árbol deben pertenecer al mismo tipo de registro, mientras que en los segundos cada nudo puede pertenecer a un tipo de registro distinto (generalmente se mantiene al mismo tipo de registro en cada nivel).

Según estas definiciones, el árbol de la empresa utilizado anteriormente pertenece a los de tipo heterogéneo ya que por ejemplo, los nudos del nivel-2, están asociados a registros de tipo departamento, los de nivel-3 a registros de tipo sección y los de nivel-4 a registros de tipo empleado.

Un ejemplo de árbol homogéneo podría ser uno en el que se reflejara la línea de sucesión de una familia real, ya que todos los nudos estarían asociados (en cualquier nivel) a un mismo tipo de registro que contendría los datos de los varones de la familia (en el caso de que en el hipotético país sólo reinaran los hombres).

Otra característica importante dentro de los árboles, sea cual sea su tipo, es la profundidad que pueden llegar a alcanzar, es decir, el número de niveles jerárquicos que incluyen. Esta puede ser fija o variable. En el primer caso, cualquier camino entre la raiz y una variable del árbol puede tener un determinado número máximo de nudos, en cambio en el segundo no se fija ningún tope de niveles dentro del árbol. Los programas incluidos en el software básico de bases de datos, suelen estar especializados en el manejo de árboles heterogéneos, pero de profundidad fija.

1.3 Ficheros jerárquicos.

Se llama fichero jerárquico a todo aquel que permite establecer relaciones de tipo árbol entre sus registros. El manejo de este tipo de árboles no puede ser realizado nunca mediante un único fichero pla-

no, por su propia naturaleza. Por lo tanto, la única forma de gestionar ficheros jerárquicos es utilizando el software de bases de datos jerárquicas.

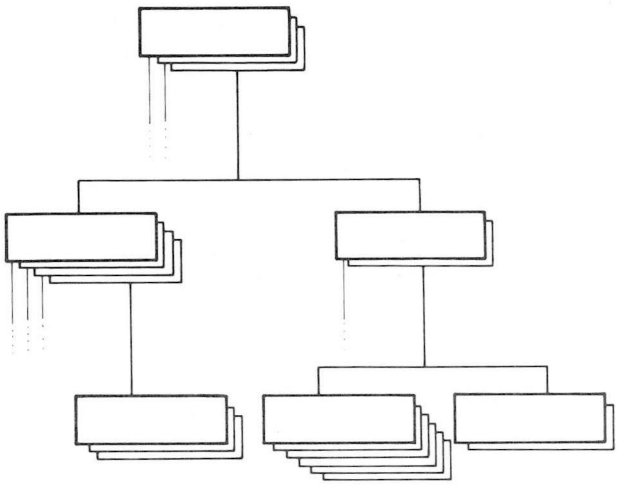


Figura 96. Un fichero jerárquico es aquel que permite establecer entre sus registros relaciones de tipo arborescente. Es decir, consiste en un árbol en el que se sustituyen los nudos por registros.

El problema principal que presentan los ficheros jerárquicos es la obligatoriedad de que cada hijo no pueda tener más que un único padre. Aunque parezca una aberración desde el punto de vista humano, es muy frecuente que en determinados tipos de organizaciones se requiera que un mismo registro tenga dos padres distintos. Por ejemplo, si se pretende representar una organización en la que se estructure la utilización de recursos materiales por los distintos grupos de una empresa, y se desea que los recursos de la empresa sean las hojas del árbol, el recurso ordenador probablemente sea utilizado por más de un grupo de la empresa y por lo tanto debería tener tantos padres como grupos utilicen el centro de cálculo. En este caso, la estructura de árbol resulta insuficiente para representar la organización y habría que recurrir a otro modelo distinto que permita que un hijo tenga más de un padre. Según este sentido, la utilización de archivos jerárquicos se encuentra muy limitada.

No obstante a la anterior limitación, la mayoría de las estructuras que representan una organización empresarial son susceptibles de representar mediante un árbol y, por lo tanto, de utilizar para su al-

macenamiento ficheros jerárquicos.

El Data Base Task Group de CODASYL utiliza frecuentemente una relación que denomina conjunto. CODASYL define: "conjunto es un árbol en el que los registros padres se consideran como propietarios y los registros hijos se consideran como miembros". En este caso, cada tipo de registros tiene un nombre, y fichero jerárquico con varios niveles puede definirse como una colección de varios conjuntos múltiples.

1.4 Bases de datos jerárquicas.

Los sistemas de bases de datos en general y los modelos jerárquicos en particular, se diseñan para facilitar a los usuarios la instalación y el manejo de grandes volúmenes de datos. Estos datos suelen estar organizados en registros, que a su vez se dividen en segmentos.

Si empezamos definiendo "segmento" como el conjunto de datos de un determinado tamaño que contiene la información de la base, y definimos "registro" como un conjunto de segmentos jerarquizados, de forma que la relación entre los distintos segmentos de un registro será alguna de las relaciones posibles en las estructuras de tipo árbol, podremos definir "base de datos jerárquica" como un conjunto ordenado de registros.

Evidentemente, la representación lógica más comunmente utilizada para agrupar la información de una base de datos jerárquica, son los ficheros jerárquicos. No obstante, como toda organización lógica, se pueden utilizar otros sistemas distintos.

Probablemente, el modelo de bases de datos jerárquicos no sea el más adecuado para representar la mayoría de las organizaciones empresariales. De hecho, en próximos apartados veremos como el modelo relacional es muy superior desde un punto de vista lógico. En cualquier caso son los modelos más utilizados hasta la actualidad, tal vez por el escaso número de recursos que consume y por todo el soporte técnico que se tiene acumulado.

Es importante no confundir las bases de datos jerarquizadas con los sistemas de ficheros tradicionales que simulan con eficiencia su funcionamiento, mediante la utilización de ficheros invertidos. En algunos casos la potencia de estas pseudo-bases de datos es suficiente para resolver los problemas de una mediana organización, pero están muy distantes de ofrecer todas las ventajas incorporadas por una auténtica base de datos.

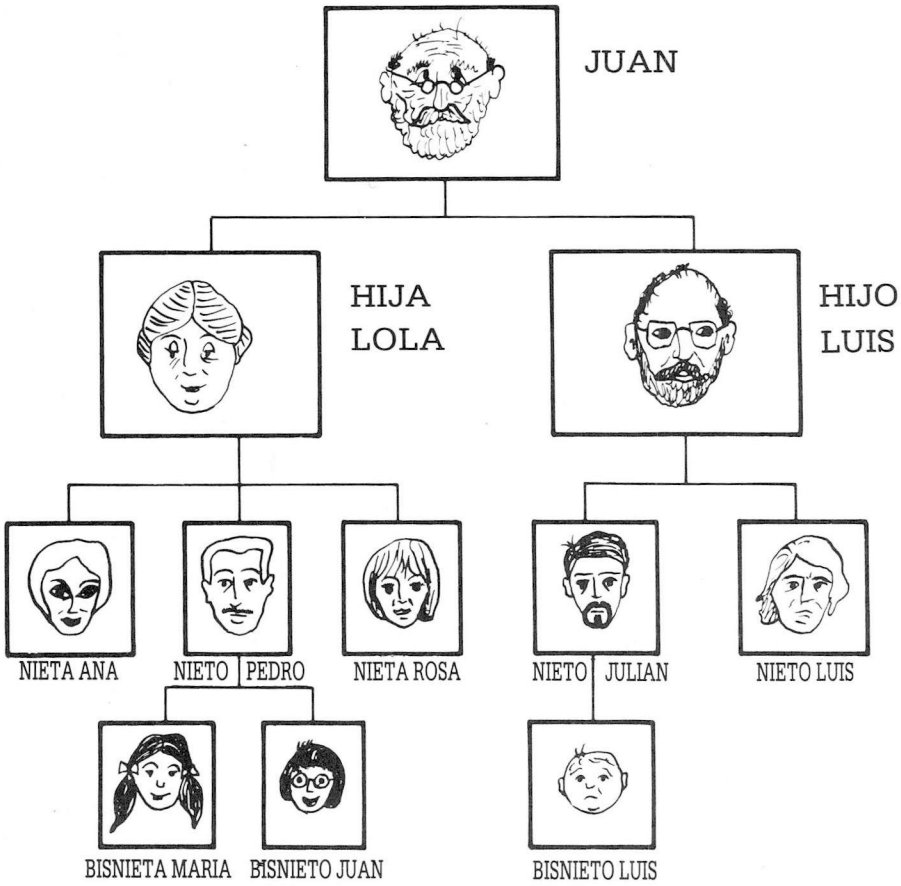


Figura 97. Uno de los ejemplos más claros de estructura arborescente es la secuencia de descendientes directos de una persona.

2.– MODELO DE RED

Mediante la utilización de estructuras jerárquicas o de tipo árbol, no es posible utilizar nudos que tengan más de un padre. Para solucionar este problema se dispone de la estructura red que sí permite esta propiedad. Por lo tanto, podemos definir base de datos modelo red, como aquella en que la relación entre los segmentos puede ser libremente definida, llegando a formar una auténtica red de registros.

No se puede afirmar que este tipo de estructura sea más potente que las arborescentes. Simplemente son distintas. En algunos ca-

sos, utilizando redes se puede aumentar mucho la similitud entre la base de datos y la organización real pero a cambio, la gestión de los datos es más complicada ya que se obtiene una representación menos clara.

2.1 Red.

Podemos definir red, como una estructura de nudos en la que pueden existir conexiones entre cualquier pareja de componentes.

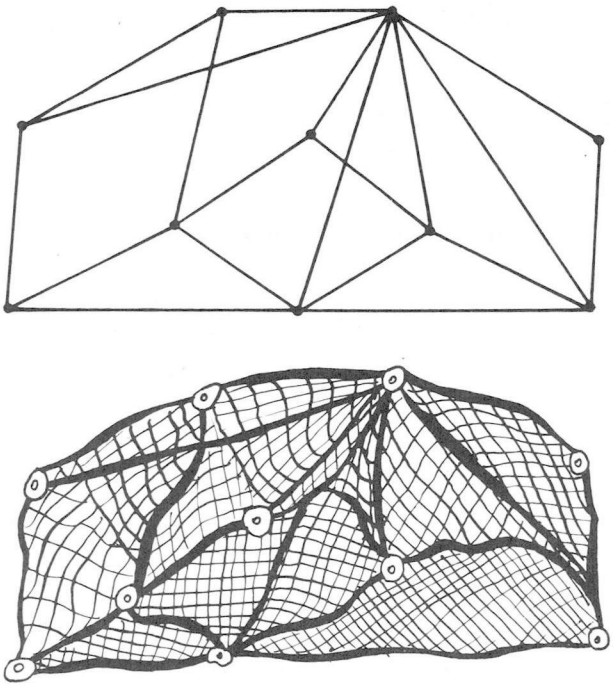


Figura 98. El hecho de denominar red a la estructura de datos que permite establecer cualquier relación entre los nudos, está basado en la similitud de dicha estructura con una red en términos generales.

Como en el caso de los árboles, la relación existente entre los elementos de una red se puede establecer en términos de padres e hijos. Pero la estructura obtenida no es tan jerarquizada como en el caso anterior. Para conseguir representar gráficamente la relación entre los distintos nudos de una red, se sitúan de forma que los nudos padres están siempre más altos que sus hijos. También es posi-

ble utilizar el concepto "nivel" dentro de las estructuras de la red, aunque no siempre como ocurría con los árboles.

A continuación vamos a describir una organización que necesariamente debe ser representada mediante una estructura de red, aunque también podría haberse utilizado una base de datos jerárquica para su implementación: supongamos que una determinada empresa utiliza cinco tipos de productos A, B, C, D y E para fabricar tres objetos distintos F, G y H que posteriormente venderá. Se trata de representar en una estructura la relación entre los productos de entrada y los objetos de salida, teniendo en cuenta que la empresa trata con dos proveedores distintos P1 y P2, el primero de los cuales suministra los productos A, B, C y E, y el segundo A, C, D y E. También hay que considerar que la empresa vende los objetos fabricados a tres clientes C1, C2 y P2. A C1 le vende los objetos F, G; a C2 los objetos F, G y H y a P2, que además de cliente es proveedor, le vende el objeto H. Para terminar la descripción del proceso tenemos que indicar que en la confección del objeto F se utilizan los productos A, B y D, en la del objeto G se utilizan los productos C y E, y en la de los objetos H se utilizan los productos B y E.

Una vez definidas todas las características de la organización a representar, podemos utilizar la siguiente estructura de red:

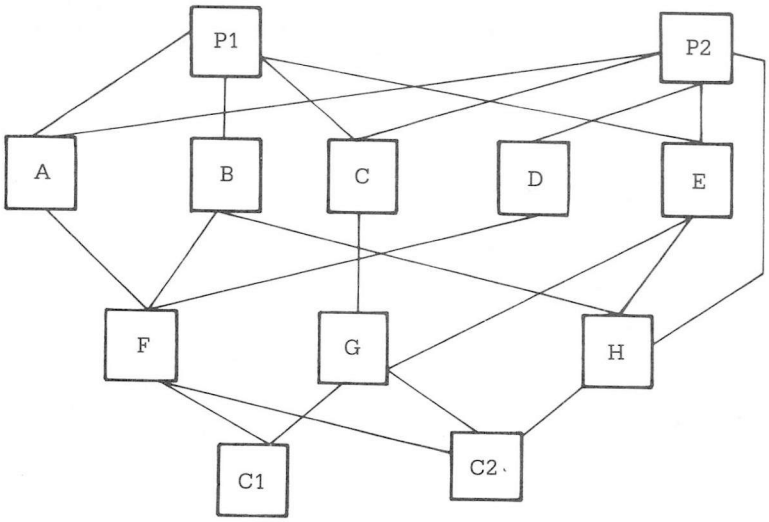


Figura 99.

Se puede observar en esta figura, cómo el objeto H tiene dos padres: los productos B y E. Incluso el cliente C2 llega a tener tres padres distintos; los objetos F, G y H. También es notorio que la relación entre los nudos no es necesariamente descendente, por ejemplo la relación entre el objeto H y el cliente (también proveedor) P2 es del sentido ascendente.

En resumen, las estructuras red permiten establecer relaciones más complejas entre sus elementos que las estructuras arborescentes, aunque esto no implica necesariamente un beneficio, puesto que la diferencia en el coste de mantenimiento puede ser superior a los beneficios obtenidos.

2.2 Tipos de redes.

Al igual que en el caso de los árboles, podemos establecer una clasificación atendiendo a los distintos tipos de registros asociados a sus nudos.

Cuando todos los nudos se corresponden con registros del mismo tipo se dice que la red es homogénea, en caso contrario, es decir, cuando se representan en la misma red al menos dos tipos de registros distintos, se dice que la red es heterogénea.

El ejemplo de la organización de la empresa dedicada a la compra de productos y venta de objetos elaborados con dichos productos, corresponde a una red heterogénea, ya que dentro de ella habría cuatro tipos de registros distintos: PROVEEDORES, PRODUCTOS, OBJETOS y CLIENTES.

Un sencillo ejemplo de red homogénea sería la representación de las descendencias de una familia en la que se tuvieran en cuenta tanto al padre como a la madre (en caso contrario, es decir, si sólo se tuviera en cuenta al padre, se trataría de un árbol).

2.3 Principales dificultades en el manejo de redes.

Ya hemos dicho que el mantenimiento de una estructura de red puede ser bastante complicado. Esto se debe básicamente a la posibilidad de que las relaciones existentes entre los nudos estén tan imbricados que dificulten la búsqueda de caminos dentro de la red. En este apartado vamos a describir dos casos en los que la dificultad queda patente: los ciclos y los lazos.

Se dice que una red incluye un ciclo cuando un nudo tiene como descendiente a un antepasado, produciéndose por lo tanto un ciclo al intentar recorrer la zona de la red que incluya a este nudo. Un

ejemplo de ciclo lo teníamos en el ejemplo anterior: el nudo correspondiente al objeto H tenía como descendiente al cliente P2, que a su vez era un antepasado suyo, ya que el nudo P2 suministraba el producto E necesario para la fabricación de H.

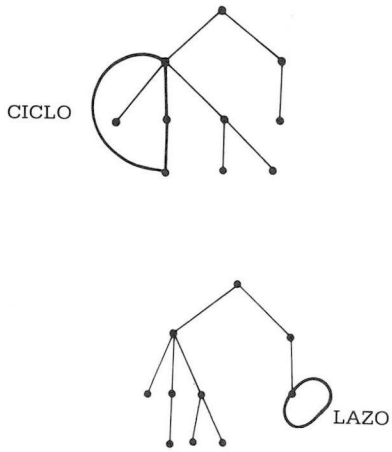


Figura 100. Uno de los principales problemas en el manejo de redes estriba en la posible existencia de ciclos y lazos. Como se puede apreciar en la figura, un ciclo consiste en la existencia de un descendiente que a la vez sea progenitor, mientras que un lazo consiste en un nudo que a su vez es descendiente de sí mismo.

Cuando en la misma red se incluyen muchos ciclos, la estructura general puede llegar a ser extremadamente compleja. Algunas bases de datos del modelo de red están preparadas para tratar ciclos, en cambio otros exigen la ausencia total de ciclos.

Un tipo especial de ciclo es cuando el mismo nudo es padre e hijo de sí mismo. En este caso se dice que se trata de un lazo. La presencia de lazos dentro de una red complica su manejo aún más que los ciclos, y muchos de los softwares de bases de datos tipo red no están preparados para tratar lazos, aún siendo capaces de manejar ciclos.

Bien para reducir estos problemas técnicos o bien simplemente para simplificar su estructura, toda red puede ser reducida a formas más simples, aunque a cambio de esta simplificación se debe introducir una cierta redundancia. En la fase de diseño de la base de datos se debe evaluar si la introducción de redundancia adicional puede ser soportada por el sistema y decidir si las mejoras conseguidas con la mayor sencillez de la red son lo suficientemente importantes

como para aumentar la redundancia.

La forma de eliminar complejidad en una red no puede ser descrita sistemáticamente. Cada caso es distinto y la única forma de representar redes de la forma más simple es utilizar un diseñador experto. Como ejemplo de simplificación de una red vamos a utilizar una parte de la red del ejemplo anterior. Supongamos que sólo consideramos proveedores y productos suministrados; en ese caso la red quedaría de la siguiente manera:

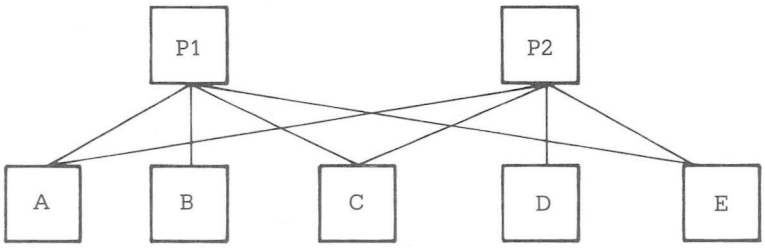


Figura 101.

Evidentemente, la mayor dificultad de esta mini-red estriba en el hecho de que haya hijos con más de un padre, como el producto A que tiene como padres a ambos proveedores P_1 y P_2. La forma de eliminar esta dificultad es sencilla: vale con invertir el orden de los nudos, con lo que quedaría:

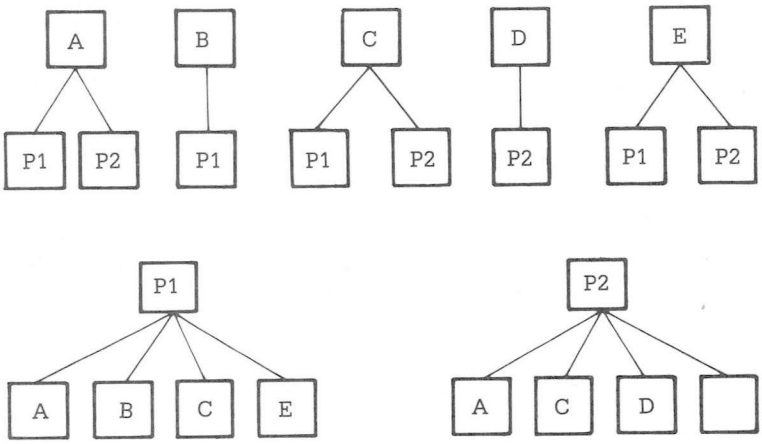


Figura 102.

Se puede comprobar cómo la sencillez estructural es conseguida mediante la inclusión de una mayor redundancia. La difícil cuestión que queda abierta es: ¿merece la pena este aumento de redundancia?. En algunos casos la respuesta es contundentemente afirmativa; esto sucede cuando la mejora conseguida hace que el tratamiento de la red sea posible mediante un software de base que no podría manejarla de otra forma. En cambio, la respuesta no es tan clara cuando los objetivos de la simplificación son meramente lógicos.

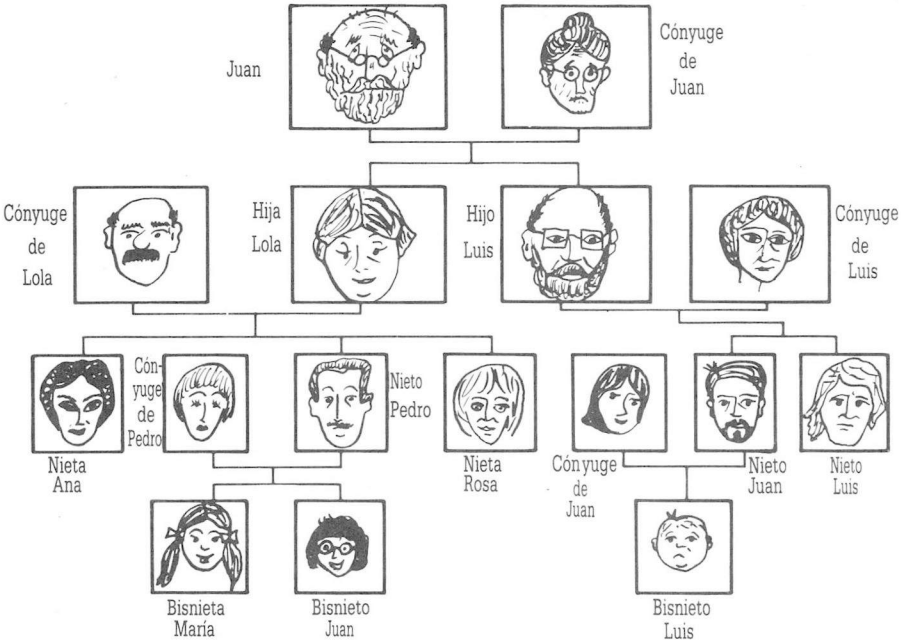


Figura 103. Una relación imposible de representar mediante un árbol, es la existencia de más de un único padre para cada nudo. Si se desea como en la figura, representar la secuencia de descendientes de una persona incluyendo los cónyuges, es imprescindible utilizar una estructura de red.

2.4 Bases de datos en red.

La razón fundamental por la que intentábamos simplificar la estructura de la red convirtiéndola prácticamente en un árbol, era para facilitar su almacenamiento físico. Algunos software de bases de datos en red, trabajan más fácilmente con árboles (no olvidemos que un árbol es una red simplificada). La principal diferencia entre las ba-

ses jerarquizadas y las de red estriba en los tipos de esquemas que admiten.

En definitiva, las bases de datos modelo red no son muy distintas de las jerárquicas, ya que de alguna manera se pueden considerar como una generalización de las anteriores. No obstante, sus cimientos teóricos están bastante diferenciados.

3.– MODELO RELACIONAL

Tanto en el modelo jerárquico como en el modelo de red, una base de datos debe ser descrita mediante una estructura lógica que acabará pareciendo una enorme maraña de nudos y que dificultará notablemente su interpretación. En la mayoría de los casos, las ideas más sencillas suelen ser las mejores además de las más simples de entender. Para evitar el enmarañamiento y conseguir descripciones sencillas de las bases de datos, E. F. Codd recomendó una serie de normalizaciones que al ser aplicadas sobre árboles y redes, garantizaban la ausencia de determinadas situaciones engorrosas. Estas normalizaciones dieron lugar a un nuevo modelo de bases de datos que nada tenía que ver con los anteriores: el *modelo relacional*.

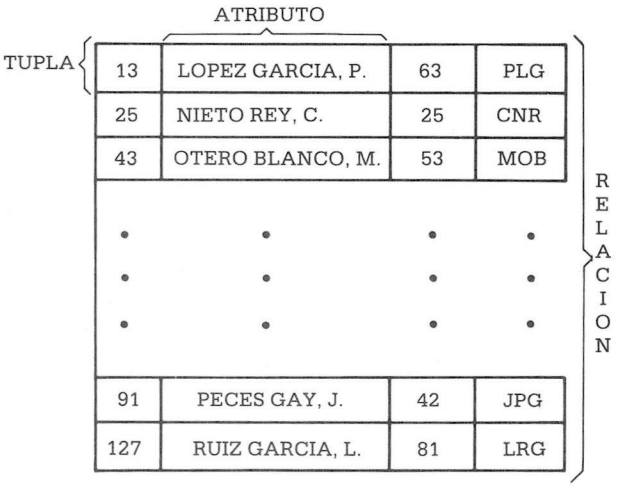


Figura 104. Si comparamos esta figura con la figura 80, podemos observar que las únicas diferencias entre relación y fichero plano estriban en la terminología utilizada. La ventaja incorporada por la relación, consiste en la flexibilidad de una base de datos relacional para que cada usuario defina sus propias relaciones.

Como ya hemos repetido varias veces, las bases de datos pretenden conseguir la independencia entre la información y el software y hardware en que se procesa. Para conseguir que esta independencia sea lo más productiva posible, se debe lograr que tanto usuarios como programadores entiendan fácilmente la descripción de los datos. También es importante que las puertas para el crecimiento de los datos queden abiertas tanto cualitativa como cuantitativamente. Además, es deseable que la base pueda ser utilizada para realizar procesos no previstos. Estas características se veían frenadas por la complejidad de los árboles o redes de los dos modelos anteriores. Para superar esta complejidad el modelo relacional trabaja con estructuras mucho más sencillas: simples tablas bidimensionales, también denominadas relaciones.

3.1 Relación.

La manera más usual para representar datos dentro y fuera del ámbito informático es la tabla bidimensional. Tanto los usuarios como los programadores están familiarizados con este tipo de organización, por lo que el esfuerzo necesario para interpretar su significado es mínimo. Al igual que toda red puede expresarse mediante uno o varios árboles, toda estructura organizativa de una base de datos puede expresarse en forma de relaciones, admitiendo de nuevo una adición de redundancia. Estas ampliaciones de redundancia, cada vez que se facilita más la interpretación de los datos, hacen que los ordenadores utilizados para soportar modelos relacionales deban ser muy potentes.

Con objeto de no perder ninguna de las relaciones existentes entre los datos, al ser representados en forma de tablas, la organización de las relaciones debe hacerse paso a paso, siguiendo unas normas dadas por Cood, hasta completar el proceso de normalización. Las propiedades fundamentales que deben verificar las relaciones integrantes de la base de datos son las siguientes:

1.– Todos los datos situados en una misma columna de la tabla deben ser homogéneos, es decir, del mismo tipo.
2.– Cada elemento de la tabla es simple y por lo tanto, en una fila y columna determinadas no puede haber más que un único elemento (nunca una nueva tabla).
3.– Todas las filas de las tablas tienen al menos un elemento diferente. Por lo tanto, no pueden existir dos filas iguales dentro de una misma relación.
4.– Cada columna debe tener un nombre propio que permita

identificar el significado de su contenido.

5.– El recorrido, tanto de filas como de columnas, se puede realizar en cualquier secuencia, sin que ello modifique el contenido de la tabla.

3.2 Terminología.

Si dentro de la teoría de la información no numérica (bases de datos), la terminología no está muy fijada dado lo novedoso de esta ciencia, dentro de la parte que estudia las estructuras relacionales se ve aumentada la confusión en la forma de denominar los principales conceptos. A continuación vamos a intentar fijar la terminología más extendida a la vez que introduciremos al lector en las estructuras relacionales:

1.– Relación.
Dados n conjuntos $D_1, D_2, ..., D_n$, llamados dominios (no necesariamente distintos), una relación R sobre ellos es un subconjunto del producto cartesiano $D_1 \times D_2 \times ... \times D_n$.

2.– Grado de relación.
Se llama grado de la relación al número n de conjuntos en que se basa.

3.– Dominio.
Nos referiremos a Di como el i-ésimo dominio de la relación.

4.– n-Tupla.
Se llama n-tupla o simplemente tupla, a cada uno de los elementos de una relación de grado n.

5.– Cardinalidad de la relación.
Se llama cardinalidad de la relación R, al número de n-tuplas que contiene.

6.– Atributos.
Dado que los dominios de una relación no son necesariamente disjuntos, llamamos atributo a cada uso particular de un dominio.

Veamos, mediante un ejemplo, una sencilla representación de todos estos conceptos. Supongamos una relación de proveedores en la que se quieren incluir un código identificador (P#), el nombre del proveedor (NOM-P), la ciudad de residencia del proveedor (CIUDAD), la dirección del proveedor (DIR-P) y el distrito postal (POSTAL), y supongamos que a la relación se la denomina PROV y dispone de los siguientes datos:

P #	NOM-P	CIUDAD	DIR-P	POSTAL
1	JULIAN	MADRID	C/ PEZ, 13	15
5	ANTONIO	TOLEDO	C/ OCA, 2	12
9	PEDRO	MADRID	C/ LUNA, 6	15
7	RAMON	MADRID	C/ RAIZ, 9	9
13	CARLOS	SEGOVIA	C/ FLOR, 2	16

Figura 105.

El primer dominio D de los identificadores está formado por el conjunto de los números naturales, al igual que el dominio D de los códigos postales. Este es un caso en el que se observa cómo los dominios no tienen por qué ser disjuntos. El grado de la relación PROV es 5, dado que se basa en cinco dominios que dan lugar a los atributos: P#, NOM-P, CIUDAD, DIR-P y POSTAL. Una 5-tupla de la relación PROV puede ser: <1, JULIAN, MADRID, C/ PEZ, 13, 15> y la cardinalidad de la relación es 5, ya que está compuesta por 5 tuplas.

3.3 Identificador de tupla.

Toda tupla de una relación debe contener algún atributo que permita su identificación. A estos atributos se les denomina identificadores de tupla. En el ejemplo anterior existían dos atributos candidatos a identificador de tupla: P# y NOM-P. Evidentemente, cualquiera de los dos permite acceder a una única tupla de forma segura. Por razones de comodidad parece más lógico elegir a P # como identificador. No obstante, NOM-P podría resultar un identificador igualmente válido.

La condición que debe verificar un atributo para ser identificador de tupla es la siguiente:

- Debe permitir la identificación unívoca de la tupla a partir de un valor concreto suyo.

Aunque en el caso anterior un único atributo podría hacer las veces de identificador de tupla, en otros casos es necesario utilizar más de un atributo para obtener un identificador. Por ejemplo, si suponemos una relación de pedidos por proveedores en la que se incluyen tres atributos: Identificador del proveedor (P#), identificador del artículo pedido (A#) y número de unidades pedidas (CANTIDAD), la única forma de identificar plenamente una tupla será mediante los atributos P # y A# ; así pues, el identificador de tupla en esta relación será el conjunto de dos atributos: P # y A #.

Cuando en una relación existen varios atributos que pueden desempeñar la labor de identificador, como en el ejemplo de la relación de proveedores lo eran P# y NOM-P, se les denomina candidatos a identificadores y cuando se elige uno entre los candidatos a identificadores, por ejemplo P#, se le pasa a denominar identificador primario.

3.4 Cálculo relacional.

Una de las mayores ventajas del modelo relacional, estriba en la sencillez con que pueden ser manipuladas las relaciones. Cuando trabajábamos con árboles o redes resultaba verdaderamente complicado operar con ellos, en cambio con una o varias relaciones resulta sorprendentemente fácil realizar cualquier operación. Más adelante, cuando estudiemos los lenguajes de manipulación de bases de datos relacionales, veremos con detalle todos los operadores del cálculo relacional. Ahora vamos a limitarnos a practicar con las operaciones más sencillas.

Una de las operaciones más necesarias es la que permite la eliminación de determinadas columnas de una relación. A esta operación se le denomina proyección y se le nota mediante el símbolo Π. La forma de invocarla es la siguiente:

NOMBRE-NUEVA-RELACION Π NOMBRE-ANTIGUA-RELACION
(ATRIBUTOS DE LA ANTIGUA RELACION)

y el resultado producido en la nueva relación, está compuesto por un conjunto de tuplas distintas formadas por los atributos solicitados.

Veamos un ejemplo:

Sea la relación CLIENTES, cuyas tuplas contienen un identificador de cliente (C#), el nombre del cliente (NOM-CLI), la ciudad en que habita (CIUDAD), la provincia (PROV) y la tarifa de facturación que se le aplica (TARIFA), con los siguientes datos:

CLIENTES

C #	NOM-CLI	CIUDAD	PROVINCIA	TARIFA
1	BLANCO	MADRID	MADRID	27
3	GARCIA	POZUELO	MADRID	32
9	SANCHEZ	CALATAYUD	ZARAGOZA	16
12	PEREZ	MADRID	MADRID	27
17	LOPEZ	ARANJUEZ	MADRID	16
19	VILA	ZARAGOZA	ZARAGOZA	32

Figura 106.

Si sobre esta relación hacemos la siguiente proyección:

RESUMEN ↑↑ CLIENTES (NOMB-CLI, CIUDAD) el resultado es el siguiente:

RESUMEN

NOM-CLI	CIUDAD
BLANCO	MADRID
GARCIA	POZUELO
SANCHEZ	CALATAYUD
PEREZ	MADRID
LOPEZ	ARANJUEZ
VILA	ZARAGOZA

Figura 107.

Si sobre la misma relación inicial proyectamos el atributo provincias, el número de tuplas producido será menor a las existentes en la relación incial, ya que algunas de ellas están repetidas. Así si realizamos:

PROVINCIAS ↑↑ CLIENTES (PROV), el resultado es:

PROVINCIAS

PROV
MADRID
ZARAGOZA

Figura 108.

Evidentemente, esta potente operación facilita mucho la explotación de la base por los programadores, e incluso por los usuarios finales, ya que les permite definir sus propias relaciones a partir de una dada.

La operación contraria a la proyección (que en el fondo se encarga de dividir una relación) es la unión, cuyo cometido es reunir dos relaciones en una sóla. El símbolo utilizado para representar a este operador es el *, y para que dos relaciones sean susceptibles de reunirse, se deben cumplir algunas condiciones entre las que destaca la existencia de un atributo común por el que se pueda efectuar la unión. Al igual que en el caso de la proyección, al unir dos relaciones no es necesario que la relación resultante contenga todos los atributos de ambas; para especificar esta característica se utiliza el siguiente formato:

NOMBRE-NUEVA-RELACION = NOMBRE-RELACION-A (ATRIBUTOS DE A) * NOMBRE-RELACION-B (ATRIBUTOS DE B)

La relación resultante contendrá todas las tuplas que surjan de la unión de los atributos especificados de las relaciones A y B.

Veamos un ejemplo:

Sea la relación PERSONAL con los atributos: Número de empleado (N#), nombre del empleado (NOM), departamento del empleado (DEP) y categoría (CAT), y sea la relación CATEGORIAS con los atributos: clave de categoría (C#) y sueldo correspondiente a la categoría (SUELDO). Supongamos que ambas relaciones tienen las siguientes tuplas:

PERSONAL

N.° #	NOMBRE	DEP.	CAT.
1	PEREZ	17	1
3	GARCIA	19	2
5	MORENO	17	2
9	MARTIN	21	3
13	LOPEZ	17	1
15	REY	21	2

CATEGORIAS

C #	SUELDO
1	1.000.000
2	1.500.000
3	2.000.000

Figura 109.

Si ejecutamos la operación unión con las siguientes características:

NUEVA = PERSONAL (NOM) * CATEGORIAS (SUELDO),

el resultado sería el siguiente:

NUEVA

NOMBRE	SUELDO
PEREZ	1.000.000
GARCIA	1.500.000
MORENO	1.500.000
MARTIN	2.000.000
LOPEZ	1.000.000
REY	1.500.000

Figura 110.

Los dos operadores descritos ofrecen muchas posibilidades, pero no representan más que un ejemplo de todas las operaciones que se pueden realizar con relaciones. En realidad existe una disciplina llamada cálculo relacional, que se encarga de estudiar formalmente las posibles operaciones sobre relaciones. Se puede llegar a hablar de un álgebra relacional formada por el conjunto de todas las posibles relaciones y las operaciones, que aplicadas a una o más relaciones producen otras nuevas. La forma en que el usuario puede obtener el resultado deseado, es mediante la ejecución de una secuencia de operaciones.

El tratamiento dado en este apartado al cálculo relacional, resulta algo superficial. En próximos capítulos lo estudiaremos más detalladamente, no obstante ya se pueden apreciar algunas ventajas presentadas por este soporte teórico:

1.– Se pueden definir ciertas propiedades de datos y comprobar, automática y continuamente, si se verifican. De esta forma se establece un procedimiento cómodo de seguridad.

2.– El usuario se despreocupa de los métodos utilizados por el ordenador para ejecutar sus operaciones. A él le vale con estudiar las operaciones del cálculo relacional.

3.– La forma en que el usuario plantea sus preguntas mediante el cálculo relacional, es muy similar a su razonamiento anterior a la mecanización. Tan solo modifica el lenguaje.

3.5 Bases de Datos Relacionales.

Se puede definir base de datos relacional como un conjunto finito de relaciones variables en el tiempo, definidas sobre un conjunto finito de dominios. Las visiones de las relaciones son distintas según las necesidades del usuario; utilizando un símil vulgar, se puede afirmar que cada usuario define sus propias relaciones "cortando y pegando" las relaciones del esquema de la base. Precisamente por la necesidad de gestionar a través del ordenador grandes volúmenes de datos, es necesario utilizar equipos potentes si se quiere funcionar correctamente con este tipo de bases de datos.

Figura 111. Una de las dos principales ventajas incorporadas con la utilización de bases de datos relacionales, es la sencillez de las ideas manejadas. Desde la niñez estamos acostumbrados a manejar continuamente relaciones para ordenar nuestros conocimientos.

La particularización de las distintas visiones lógicas de una base de datos relacional es muy cómoda; en palabras de uno de los grandes teóricos sobre este modelo, como es Cood: "Tanto el programador de aplicaciones como el usuario interactivo, ven la base de datos como una colección variable en el tiempo de relaciones normalizadas de diferentes grados". El concepto normalización ha sido definido someramente en este capítulo. En próximos capítulos estudiaremos con detalle los distintos niveles de normalización según Cood.

Existe el convencimiento general de que tanto por su sencillez como por su potencia, las bases de datos relacionales son ideales para la gestión de grandes volúmenes de datos. El único inconveniente es que el consumo de recursos es superior cuando se utiliza un modelo relacional que cuando se utilizan modelos jerárquicos o de red.

A continuación vamos a citar algunas de las principales ventajas que aporta la utilización de bases de datos relacionales:

1.– Dado que existe un soporte teórico muy fuerte en el campo de las matemáticas, mediante el modelo relacional se puede trabajar con mayor precisión que con modelos menos fundamentados.

Figura 112. La otra gran ventaja de las bases de datos relacionales estriba en el gran soporte teórico que obtienen de las Matemáticas.

2.– Las operaciones proyección y unión de relaciones, ofrecen gran flexibilidad para que los usuarios puedan definir sus ficheros lógicos.

3.– Por la sencillez de la estructura relacional, resulta mucho más sencillo entender y manejar una base relacional que otra jerárquica o de red.

4.– Resulta más fácil y fiable la implantación de sistemas de seguridad, ya que se pueden limitar los atributos de las relaciones manteniendo aparte las que resulten confidenciales.

5.– Otra ventaja de este modelo, aunque sea un tema no tratado en este libro, es la facilidad de implementación física. Los sistemas relacionales resultan mucho más adecuados a los procedimientos de gestión del almacenamiento.

En la actualidad, la tendencia dentro del mercado informático apunta hacia un abaratamiento del hardware y un encarecimiento del software. Como uno de los principales defectos de las bases de datos relaciones es la necesidad de utilizar equipos muy potentes, es de suponer que dentro de pocos años estas bases de datos se impongan al resto de los modelos.

CUESTIONARIO CAPITULO 7

MODELOS DE BASES DE DATOS

1.– ¿Qué se entiende por árbol?
2.– ¿Qué relaciones existen en un árbol?
3.– ¿Qué tipos de árboles conoce?
4.– ¿Cuál es el principal problema de los ficheros jerárquicos?
5.– ¿Qué es una red?
6.– ¿Cuáles son los principales tipos de redes?
7.– ¿Qué es un ciclo y qué es un lazo?
8.– ¿Qué es una relación y qué propiedades tiene?
9.– ¿Qué es el identificador de tupla?
10.– ¿Qué ventajas tienen las bases de datos relacionales?

CAPITULO VIII

UTILIZACION DE BASES DE DATOS JERARQUICAS

1.– PRINCIPALES LENGUAJES DE DATOS

Tanto el administrador de la base de datos como los programadores deben poder describir con rigor los datos necesarios. Para ello tienen a su disposición una serie de lenguajes de datos, que consisten en el medio por el que se declaran al sistema cuáles son las estructuras que van a ser utilizadas.

La importancia de que los lenguajes de datos sean sencillos y a la vez potentes, es tan vital, que los fabricantes de ordenadores han dedicado mucho tiempo y dinero en facilitar la labor de descripción de los datos. El resultado de estos esfuerzos han sido programas de utilidad que han incorporado a sus softwares de base.

Dentro de los lenguajes convencionales de datos, el más extendido es sin duda el DL/I, del fabricante IBM. No obstante, intentando conseguir un lenguaje no ligado a un fabricante determinado, se han creado grupos independientes para el estudio de estos temas. Uno de los grupos más importantes es la comisión creada por CODASYL para trabajos en lenguajes de datos.

En este capítulo vamos a estudiar precisamente estos dos lenguajes de datos, con lo que podremos obtener una visión de conjunto de las distintas formas en que puede ser utilizada una base de datos jerárquica.

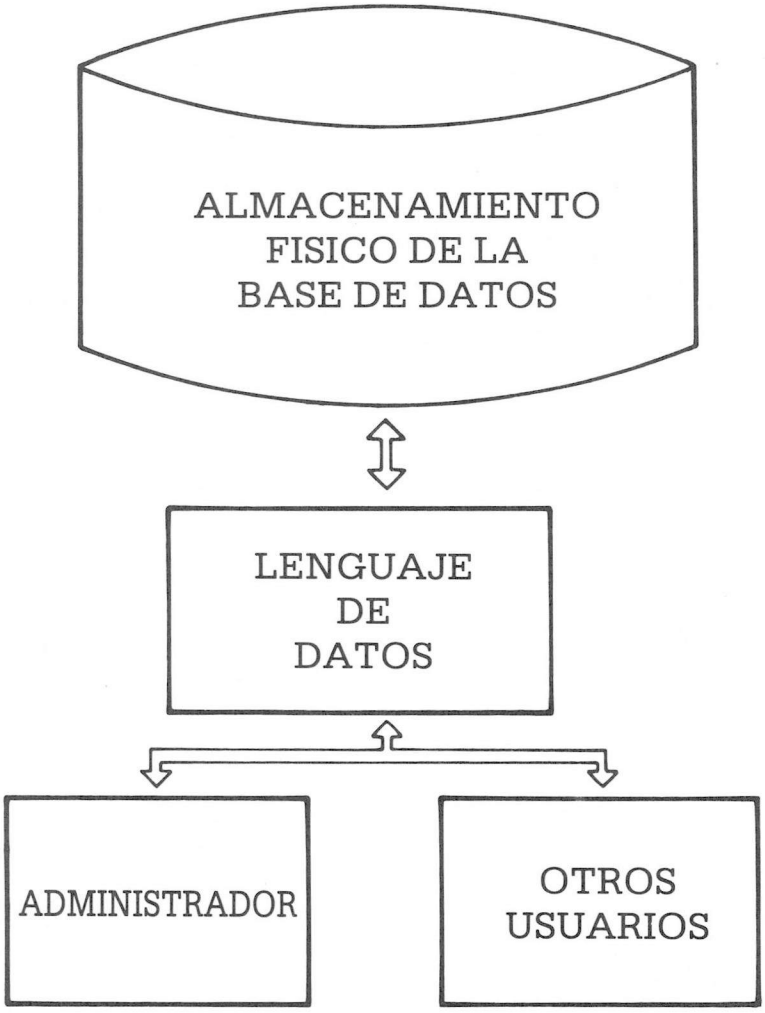


Figura 113. Tanto el administrador de la base como el resto de los usuarios se valen del lenguaje de datos para poder describir y utilizar la información contenida en la base de datos.

1.1 Objetivos de los Lenguajes de Datos.

Para conseguir una descripción lógica adecuada de los datos, los lenguajes encargados de ello deben cumplir una serie de objetivos, algunos de los cuales son los siguientes:

1.– Deben definir claramente las posibles subdivisiones de los

datos. Cada lenguaje se diferencia de los demás por las posibilidades de subdivisión.

2.– Deben asignar un nombre único a cada una de las subdivisiones de datos. La subdivisión de mayor nivel será la base de datos en su totalidad y por consiguiente se le deberá asociar un nombre. Las subdivisiones de menor nivel serán los datos elementales que también deberán tener un único nombre, y todas las subdivisiones intermedias como registro, archivo, etc., también deberán ser "bautizadas" adecuadamente.

3.– Para cada subdivisión debe indicarse el tipo de datos que contiene y en qué secuencia.

4.– Se debe indicar cuáles de los datos van a ser utilizados como identificadores.

5.– Debe indicarse la relación existente entre los tipos de datos de la base y dar un nombre a dicha relación.

6.– Se puede indicar también, aunque no es obligatorio, características sobre la codificación de los programas que accederán a la base, longitud y dominio de cada dato, volumen total de cada tipo de datos, verificación de errores para cada dato, niveles de seguridad y, en definitiva, cualquier otra característica que ayude a conocer de forma exhaustiva los datos de la base.

1.2 Lenguaje DL/I.

El DL/I (Data Lenguaje I) es el lenguaje de datos potenciado por IBM para su sistema de bases de datos jerárquicos IMS (Information Management System). Sirve tanto para la definición de la representación lógica de la base, como para la definición de la representación física.

Como ya habíamos visto en el capítulo dedicado a la información, entre la representación física de los datos y la representación lógica aportada por los programas de aplicación, existen cuatro visiones de la base de datos. Dentro de los sistemas IMS, esas cuatro visiones se manejan de la siguiente forma:

1.– Visión física.
Para definir la descripción física de los datos, se puede utilizar el lenguaje DL/I en entorno DBD (Data Base Definition) físico.

2.– Visión lógica global.
Esta visión se indica en el esquema de la base de datos y puede realizarse mediante el lenguaje DL/I en entorno DBD lógico.

3.– Visión lógica parcial.
En el sistema IMS, para definir visiones lógicas parciales de la base (subesquemas), se utiliza también el lenguaje DL/I pero en entorno PSB (Bloque de especificación de programas), que a su vez está formado por unos subbloques que contienen la información correspondiente a cada programa, a los que se denomina PCBs.

4.– Visión de usuario final.
Esta visión es aportada por los programas de aplicación y por tanto se definirá mediante las instrucciones oportunas del lenguaje utilizado para la programación, al que denominaremos desde ahora: lenguaje anfitrión.

1.3 Lenguaje CODASYL.

Este lenguaje fué aportado por una comisión del grupo independiente CODASYL. Desde 1.969, en la conferencia sobre lenguajes para la descripción de datos, se ha trabajado mucho para conseguir un método de definición válido para cualquier base de datos.

El objetivo fundamental del lenguaje de CODASYL, al que ellos mismos denominan DDL (Data Description Languaje), es permitir la definición del esquema. Dado que este lenguaje sólo sirve para la descripción lógica global de los datos, es necesario complementarlo con otros lenguajes para poder manipular las cuatro visiones. Así pues, si se decide utilizar este lenguaje, las visiones de la base de datos se manejan de la siguiente manera:

1.– Visión física.
Existe un lenguaje denominado DBTG (Lenguaje de Control de Dispositivos y Medios) que permite realizar la descripción física de la base de datos.

2.– Visión lógica global.
Como ya decíamos antes, para la descripción lógica global de la base de datos, se puede utilizar el lenguaje DDL.

3.– Visión lógica parcial.
Para realizar la descripción de los subesquemas se pueden utilizar algunas instrucciones declarativas de lenguaje de programación. Por ejemplo, la DATA DIVISION si se utiliza el lenguaje COBOL.

4.– Visión de usuario final.
Al igual que en el caso del DL/I, esta visión es aportada por los programas de aplicación.

2.– BASES DE DATOS IMS

IBM, compañía propietaria de este tipo de bases de datos, define IMS de la siguiente manera: "Es un sistema de gestión que forma parte del sistema operativo diseñado para facilitar al usuario la instalación y manejo de una o varias bases de datos, en una organización batch y/u on-line, soportando una amplia variedad de aplicaciones".

Los principales objetivos de IMS, que coinciden con los objetivos generales de cualquier sistema de base de datos, son los siguientes:

1.– Independencia de datos.
2.– Integridad de programas y bases de datos mediante una gestión completa de programas y base.
3.– Recuperación de datos a cargo del propio sistema.
4.– Estructuras de datos, físicas y lógicas, separadas.
5.– Terminales lógicos y físicos separados.
6.– Procesos concurrentes batch y on-line.
7.– Soporte de lenguajes de alto nivel.
8.– Grandes volúmenes de proceso con respuesta rápida.

La definición de IMS, dada por la propia IBM, permite observar que su filosofía consiste en plantear la base de datos como una extensión del sistema operativo; de esta forma, las fundamentales áreas en las que inciden, son la gestión de datos propiamente dicha y la comunicación de datos entre el sistema operativo y la base.

2.1 Operaciones en Bases de Datos IMS.

Sobre una base de datos jerarquizada se pueden hacer cuatro tipos distintos de operaciones:

a) Descripción de la base de datos.
b) Descripción para los programas.
c) Creación de la base de datos.
d) Gestión de la base de datos.

Si particularizamos en el sistema IMS, el contenido de cada tipo de operaciones es el siguiente:

a) Descripción de la base de datos.
La solución tradicional para la instalación de una base de datos (banco de datos), partía del estudio particularizado de las distintas aplicaciones que utilizaban datos referentes a una determinada gestión. En cada una de las aplicaciones, surgían unos ficheros que se describían y daban lugar a un banco de datos compuesto por ficheros independientes de distinto tipo, cada uno de los cuales disponía de programas de actualización

y de recuperación de datos. Incluso en el mejor de los casos, cuando se estudiaban las interrelaciones entre las distintas aplicaciones, existían datos redundantes y el nivel de integración era escaso.

La solución con bases de datos en general, y con IMS en particular, pasa por una descripción conjunta de todos los datos que se plasma en la descripción de la base de datos DBD. Esta descripción realiza una labor de coordinación entre todos los conjuntos de datos y las aplicaciones que los utilizan.

Dado que el IMS está soportado bajo el sistema operativo VS, la DBD se encuentra ubicada en una biblioteca IMS/VS que incluye las DBD's de todas las bases de datos de la instalación. La forma de crear o modificar DBD's de la biblioteca IMS/VS, es mediante la utilización de unas fichas de control que se encargan de ejecutar una serie de programas de utilidad para el mantenimiento DBD's. Mediante este procedimiento se cargan datos tales como el nombre de la base de datos, los nombres y las longitudes de los segmentos, los nombres y las longitudes de los campos, los métodos de acceso, etc.

Mediante las DBD's se descarga de responsabilidad al usuario, pasando a asumirla el propio sistema. Por ejemplo, las posibles redundancias, que aun utilizando IMS, implican una gestión especial para las actualizaciones, la DBD las refleja de tal forma que el usuario las desconoce.

b) Descripción para los programas.

Define Block Data

Bloque especificación de Programas

Aunque las DBD's contienen toda la información necesaria para la utilización de la base de datos, existe otro intermediario entre ellas y el usuario final: las descripciones para los programas (PSB). Mediante ellas se pueden realizar distintas visiones particulares de la base de datos según las necesidades de los programas de aplicación.

La ubicación física de las PSB's coincide con la de las DBD's, es decir, se almacenan en una biblioteca IMS/VS y para su carga y modificación se utiliza un método paralelo, mediante determinadas fichas de control se realiza la ejecución de los programas de utilidad para el mantenimiento de las PSB's. Cada PSB contiene su propio nombre, el nombre de la base de datos (DBD), las funciones permitidas, los segmentos sensibles, etc.

La función fundamental de las PSB's es ofrecer distintas versiones jerárquicas de la misma información y para ello especifican las características de cada visión en una PCB (bloque de

especificación de programas). Mediante las PCB's no sólo se consigue particularizar la información a gusto del usuario final (programa), sino que también se consigue salvar la modificación de los programas, ante cambios en la estructura de los datos que no le afecten.

c) Creación de la base de datos.

Una vez definida la DBD y las PSB's, estamos en disposición de crear la base de datos. El trabajo de la generación inicial de la base suele estar encomendado al administrador del sistema, el cual tendrá en cuenta las necesidades de los distintos usuarios.

Además de la biblioteca IMS/VS, con la DBD y las PSB's podrán utilizarse otros ficheros en cualquier tipo de soporte, como discos magnéticos, cintas, tarjetas perforadas, etc. La generación se realizará mediante uno o varios programas de creación que se realizarán en alguno de los posibles lenguajes anfitriones y DL-I.

En resumen, podemos afirmar que los programas de creación de la base de datos utilizarán como entrada la biblioteca IMS/VS y los soportes que contienen los datos iniciales, y como salida, generarán la base de datos propiamente dicha.

d) Gestión de la base de datos.

Los procesos de recuperación, inserción, actualización y borrado de información de la base de datos, se realizarán directamente por los usuarios mediante programas de aplicación. Estos programas redactados en el lenguaje anfitrión, invocarán a cualquiera de los procedimientos del lenguaje de consulta DL-I que a su vez, en determinados casos, leerán y/o modificarán las informaciones de la biblioteca IMS/VS y gestionarán en definitiva, la base de datos.

Las instrucciones de llamada al DL-I, difieren según el lenguaje anfitrión sea PL/I, COBOL o ENSAMBLADOR, pero básicamente todos tienen el mismo formato, ya que se basan en los mismos procedimientos descritos en el apartado anterior.

2.2 Lenguaje DL-I.

El lenguaje DL-I consiste en una serie de procedimientos que pueden ser utilizados desde el lenguaje anfitrión (COBOL, PL/I...) para acceder a la parte del registro que interese al usuario, según esté definida su PCB.

Las operaciones de acceso a la base, son las siguientes:

1.– GET UNIQUE (GU).

Permite obtener un registro dentro de la base: el primero que cumpla las condiciones especificadas. Una vez que el programa redactado en lenguaje anfitrión incluye la operación GU, el control de la ejecución pasa a ser responsabilidad de la base de datos que, a su vez, cuando finaliza la búsqueda (con éxito o fracaso, según exista o no algún registro que reuna las condiciones necesarias), devuelve el control al programa principal.

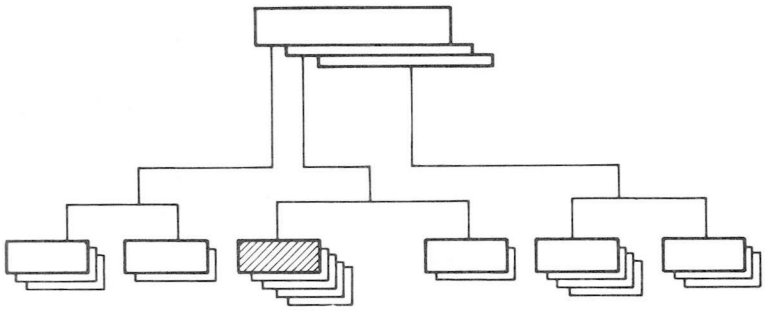


Figura 114. La operación GET UNIQUE permite obtener el primer registro de la base de datos jerárquica que cumple los requisitos exigidos. En la figura se ha marcado el supuesto primer registro rayándolo.

2.– GET NEXT (GN).

El funcionamiento de esta operación es similar al de la anterior, es decir, permite obtener un registro dentro de la base de

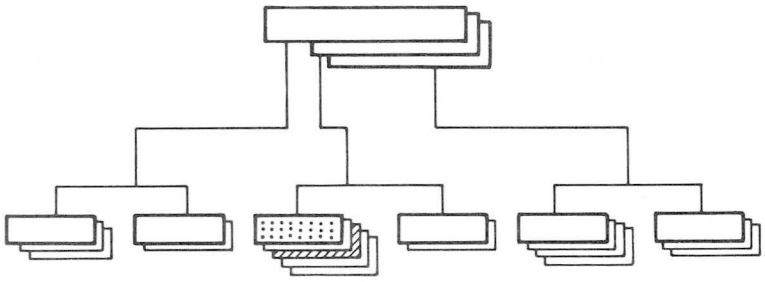


Figura 115. La operación GET NEXT localiza el siguiente registro que verifique ciertas condiciones. Suponiendo que el registro marcado con puntos fué localizado previamente con una operación GET UNIQUE, el marcado con rayas se obtendría mediante una operación GET NEXT, siempre que verifique también las condiciones exigidas.

datos que cumpla unas determinadas condiciones. La única diferencia estriba en que la GU localiza el registro dentro de la base realizando la búsqueda desde el principio, mientras que la operación GN inicia la búsqueda desde el último registro "leído".

3.– GET NEXT IN PARENT (GNP).

Este procedimiento sirve para realizar búsquedas dentro de un único registro, es decir, localiza el "siguiente" campo, dentro del mismo padre, que cumpla las condiciones exigidas.

Para su utilización es necesario haber ejecutado previamente una operación GU o GN que localice el registro donde se van a localizar los campos.

En cualquiera de los tres procedimientos que llevamos descritos, existe la posibilidad de que no se localice ningún registro o campo que satisfaga las condiciones exigidas. En este caso, se actualiza una variable con el código correspondiente a la no existencia de información para que el programa anfitrión realice las operaciones previstas para esa circunstancia.

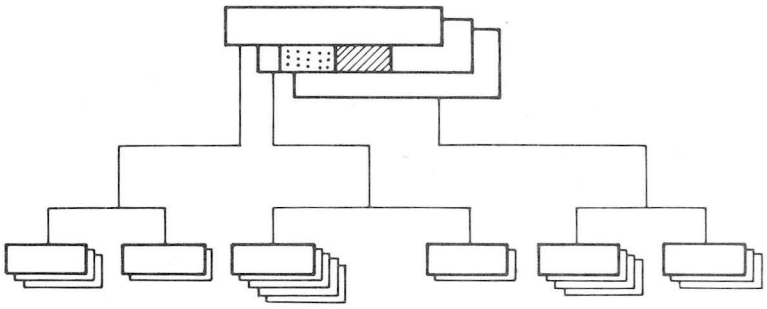


Figura 116. Mediante la operación GET NEXT IN PARENT se pueden realizar búsquedas dentro de los campos de un registro localizado previamente. Suponiendo que el campo utilizado anteriormente sea el marcado con puntos, se obtendría el campo rayado.

4.– GET HOLD (GH).

Tanto para eliminar como para actualizar un campo o registro de la base de datos, es necesario realizar previamente la localización del dato a modificar. Esto se puede realizar mediante la operación GH. Su funcionamiento es similar a los tres procedimientos descritos anteriormente. Si se desea realizar la localización para modificación del primer registro de la base

que cumpla determinadas condiciones, a las siglas GH se le añade la letra U con lo que se obtiene GHU. Si la búsqueda del registro a modificar se quiere hacer no desde el principio de la base, sino desde el último registro leído, la letra añadida es una N y la composición final quedará GHN. Por último, si se desea encontrar el siguiente campo dentro del mismo padre, la letra añadida será la P, y por lo tanto, el resultado final será GHP.

En cualquiera de los tres casos (GET HOLD UNIQUE, GET HOLD NEXT o GET HOLD NEXT IN PARENT), el resultado de la búsqueda se almacena en una zona de memoria para que el programa anfitrión realice las modificaciones oportunas.

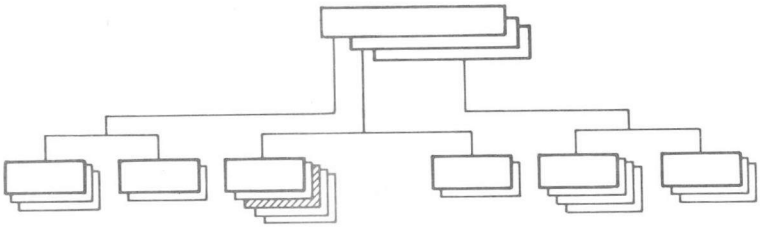


Figura 117. La operación GET HOLD sirve para localizar un registro o campo que vaya a ser modificado. Existen tres variedades dentro de esta operación: GET HOLD UNIQUE, GET HOLD NEXT y GET HOLD NEXT IN PARENT, que resultan equivalentes a las tres operaciones anteriores.

5.– INSERT (ISRT).

Este procedimiento sirve para insertar nuevos campos o registros dentro de la base de datos. Su ejecución no requiere ejecutar previamente una GET HOLD, ya que el propio sistema se encargará de incluir la nueva información en el lugar oportuno (registro o campo).

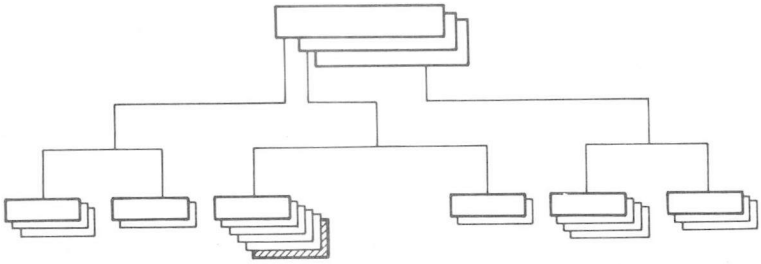


Figura 118. La operación INSERT puede ser utilizada para insertar un nuevo registro en la base de datos. En la figura se ha rayado el registro insertado mediante una hipotética operación INSERT.

6.– DELETE (DLET).

Se encarga de eliminar campos o registros ya existentes en la base de datos. Para su utilización es necesario realizar una operación GHX que localice el campo o registro a eliminar; de esta forma garantizamos que la operación de borrado se realiza sobre datos que en realidad existen en la base.

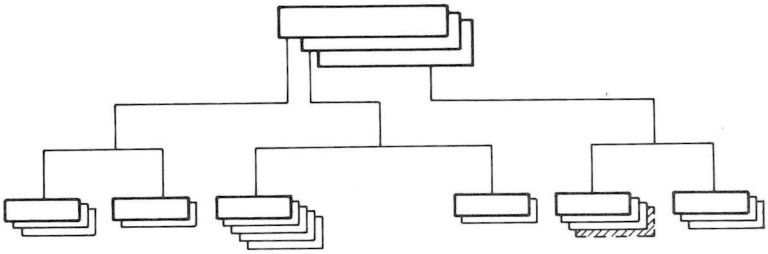


Figura 119. Al utilizar la operación DELETE se produce la eliminación del registro localizado previamente mediante cualquiera de las tres variedades de GET HOLD. En la figura se ha rayado un registro que desaparecería si se realizara sobre él una operación DELETE.

7.– REPLACE (REPL).

El cometido de esta operación es la actualización de campos o registros de la base de datos. El programa encargado de realizar la actualización, localiza la información a modificar mediante una GHX que ubica en la memoria principal los datos solicitados; a continuación, el programa anfitrión modifica los datos en la memoria principal y por último, se ejecuta la instrucción REPL que reemplaza la información inicial por el contenido de la zona de memoria actualizada.

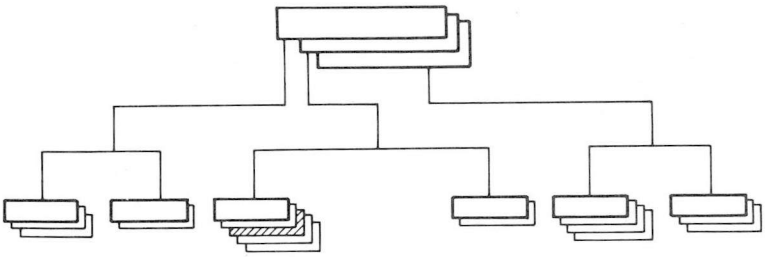


Figura 120. La operación REPLACE sirve para modificar uno o todos los campos de un registro. Como en los casos anteriores, previamente se debe haber realizado la localización correspondiente.

El programa principal es el responsable de que la actualización se ejecute correctamente y deberá, antes de invocar al procedimiento REPL, realizar los controles oportunos.

Para familiarizarnos con estas siete operaciones, vamos a realizar algunos ejercicios sobre la siguiente base de datos jerárquica. Supongamos que la base de datos debe guardar los datos del departamento de educación de una empresa. La información se estructura en forma de árbol, con un primer tipo de segmentos en los que se identifica cada curso mediante una clave (CURSO #), el nombre del curso (NOMBRE) y la descripción de las materias tratadas (DESCRIPCION). Para cada segmento curso se pueden indicar los prerrequisitos necesarios para realizarlo mediante un indicador (CURSO#) y el nombre del curso (NOMBRE) y, por otro lado, se guardarán las ofertas realizadas a los posibles alumnos, indicando la fecha en que se desarrollará el curso (FECHA), el lugar en que se impartirá (LUGAR) y el formato que se debe cumplimentar (FORMATO). Para cada una de estas ofertas habrá que indicar los profesores que intervienen en el curso, dando el identificador (EMP #) de la empresa a que pertenecen y el nombre (NOMBRE) de los profesores y, por otro lado, los alumnos matriculados, indicando para cada uno de ellos un identificador de la empresa a la que pertenece el alumno (EMP #), el nombre del alumno (NOMBRE) y la calificación obtenida (CALIFICACION).

Esta estructura se puede reflejar en el siguiente árbol:

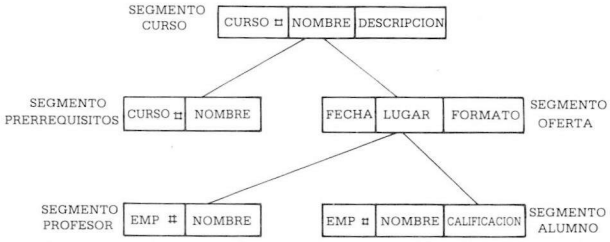


Figura 121.

1.– Para obtener la primera oferta de curso de Estocolmo haríamos:
GU CURSO
OFERTA (LUGAR = ESTOCOLMO)

2.– Para obtener todos los estudiantes, comenzando por el primero, que aparecen en la oferta anterior:

```
GU CURSO
    OFERTA (LUGAR = ESTOCOLMO)
    ALUMNO
NS GN ALUMNO
    GO TO NS
```

3.– Si se desea obtener todos los alumnos con calificación A, dentro de la misma relación anterior, haremos:

```
GU CURSO
    OFERTA (LUGAR = ESTOCOLMO)
    ALUMNO (CALIFICACION = A)
NS GN ALUMNO (CALIFICACION = A)
    GO TO NS
```

4.– Si lo que se desea es obtener todos los alumnos, sea cual sea la oferta de calificación A:

```
GU CURSO
NS GN ALUMNO (CALIFICACION = A)
    GO TO NS
```

5.– Para obtener todos los alumnos con calificación A en un curso ofrecido en Estocolmo haríamos:

```
GU CURSO
    OFERTA (LUGAR = ESTOCOLMO)
    ALUMNO (CALIFICACION = A)
NS GN OFERTA (LUGAR = ESTOCOLMO)
    ALUMNO (CALIFICACION = A)
    GO TO NS
```

6.– Para obtener toda la información de la base:

```
GU CURSO
NS GN
    GO TO NS
```

7.– Si se desean obtener los estudiantes que fueron al curso 23 que se dió el 13 de Agosto de 1.973, haríamos:

```
GU CURSO (CURSO # = 23)
    OFERTA (FECHA = 730813)
    ALUMNO
NS GNP ALUMNO
    GO TO NS
```

8.– Para obtener todos los alumnos que fueron al curso 23 y tienen calificación A:

```
GU CURSO (CURSO # = 23)
NS GNP ALUMNO (CALIFICACION = A)
    GO TO NS
```

9.– Para obtener toda la información referente al curso de código 23:

```
GU CURSO (CURSO# = 23)
NS GNP
   GO TO NS
```

10.– Si se desea introducir un nuevo estudiante para el curso de código 23 que se ofrece para el 13 de Agosto de 1984:

```
EMP# = 13
NOMBRE = PEREZ
CALIFICACION = B
INST CURSO (CURSO# = 23)
     OFERTA (FECHA = 840813)
     ALUMNO.
```

11.– Para eliminar la información del curso 23 de fecha 13 de Agosto de 1.983.

```
GHU CURSO (CURSO# = 23)
     OFERTA (FECHA = 830813)
DLET
```

12.– Para cambiar el lugar del curso 23 del que esté almacenando en la base de datos a Madrid, haríamos:

```
GHU CURSO (CURSO# = 23)
     OFERTA (FECHA = 830813)
CIUDAD = MADRID
REPL
```

2.3 Funcionamiento del IMS.

El funcionamiento del IMS puede soportar tratamientos efectuados por programas "batch" o tratamientos mediante sistemas interactivos "on-line".

El esquema de funcionamiento "batch", parte de un flujo de entrada de trabajos controlado por el sistema operativo (OS o MVS) que se encarga de gestionar a los programas de aplicación. Estos realizan su ejecución de forma normal hasta la hora de procesar una instrucción de consulta a la base de datos, en ese momento ceden el control al "sistema DL/I" que se encarga de realizar la operación indicada, devolviendo posteriormente el control al programa principal.

Por supuesto el programa puede realizar simultáneamente operaciones convencionales como imprimir informes, almacenar o recuperar datos en ficheros independientes de la base de datos, etc.

En el caso de funcionamiento "batch" no es necesaria la intervención de terminales.

Cuando el tipo de funcionamiento es "on-line", sí es necesaria la intervención de terminales que permitan al usuario final realizar las operaciones deseadas sobre la base de datos (tanto actualización de su contenido, como obtención de resultados). En este caso, el sistema aisla el efecto de un usuario del de todos los demás, es decir:

1.– Permite que dos usuarios actualicen simultáneamente la misma base de datos pero no la misma ocurrencia en un segmento.
2.– Invalida los efectos de cualquier usuario que no finalice correctamente su tarea.

En resumen, podemos afirmar que el sistema protege a cada usuario del resto de usuarios e incluso de sí mismo.

Un ordenador que se utilice para el mantenimiento de una base de datos IMS no tiene por qué estar dedicado en exclusiva a esa tarea. En un mismo momento se puede estar efectuando operaciones sobre la base de datos y ejecutando programas que no tengan nada que ver con la base.

Un ejemplo del estado del ordenador, trabajando simúltaneamente en IMS y otros programas, puede ser el siguiente:

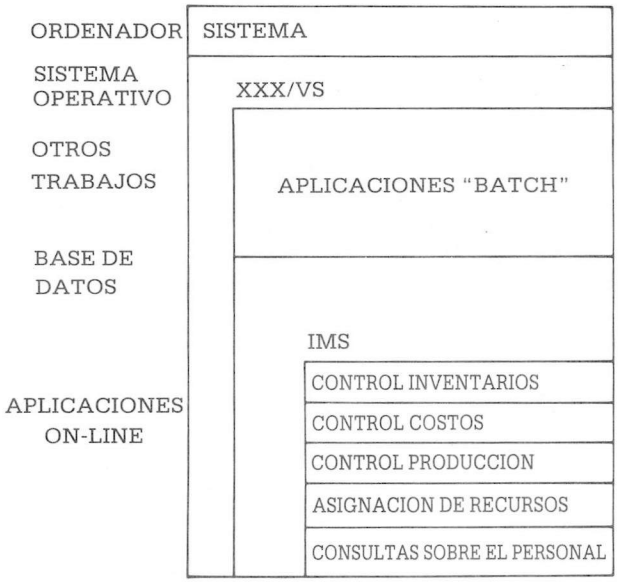


Figura 122.

Las funciones de comunicación de datos de IMS son las siguientes:

1.– Soporte de terminales físicos.

Todos los terminales conectados deben estar registrados por el IMS, que verificará la identificación del terminal cuando éste se quiera conectar con el sistema. El término terminal, está siendo utilizado en sentido amplio, es decir, consideramos que un terminal puede ser una pantalla, una impresora, un dispositivo de almacenamiento de datos, ..., o un conjunto de varios equipos.

2.– Control centralizado del sistema.

Existe un terminal maestro que controla todos los recursos del sistema:

- A nivel de comunicaciones se encarga de gestionar las líneas de comunicación, los terminales lógicos y los terminales físicos.
- A nivel de planificación controla las transacciones y programas, las regiones de mensajes y la propia base de datos.
- A nivel del sistema se encarga de controlar y arrancar copias de seguridad y recuperación de bases de datos y estado del sistema.

3.– Operaciones con terminales lógicos.

El concepto de terminal lógico sirve de nexo entre los programas de aplicación, los terminales físicos y las regiones de mensajes:

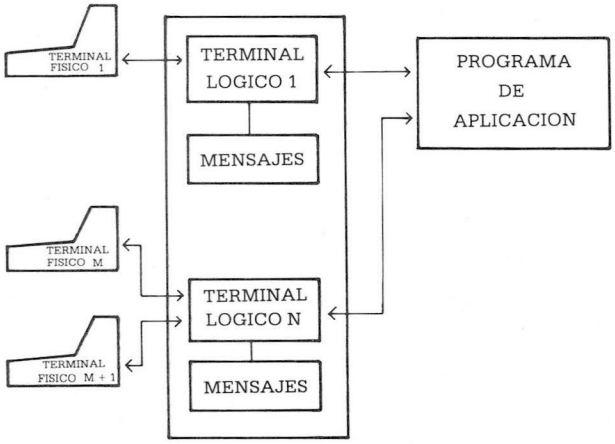


Figura 123.

4.– Proceso de mensajes.

El proceso de mensajes se realiza mediante un lenguaje de alto nivel que planifica la interrelación entre los programas y la base de datos y los procesos conversacionales.

El formato de los mensajes de entrada consta de tres zonas distintas:

1.– En la primera se almacena el código de la transacción, el número del terminal lógico o el comando.
2.– La segunda consiste en un código de seguridad.
3.– La tercera y última zona se reserva para el texto del mensaje propiamente dicho.

5.– Seguridad y confidencialidad.

Para la entrada de mensajes, mediante la segunda zona del formato, se garantiza la no utilización de la información más que por los programas o terminales autorizados.

6.– Comandos del sistema.

Mediante un conjunto de comandos del sistema se realizan cierto tipo de controles sobre la información.

Los comandos del IMS se dividen en dos grupos distintos. El primero sólo se puede utilizar desde el terminal maestro y, por lo tanto, su uso está reservado para el administrador de la base, mientras que el segundo puede ser utilizado por cualquier usuario no distinguido y desde cualquier terminal conectado al sistema bajo IMS.

3.– OTRAS BASES DE DATOS JERARQUICAS (MODELO CODASYL)

Como se podrá observar en el último capítulo de este libro, además de la IMS existen muchas bases de datos jerárquicas en el mercado informático. Desde el punto de vista práctico que pretendemos imprimir en este texto, sería ilógico pretender detallar exhaustivamente la forma en que se manejan los datos en cada una de ellas, por ese motivo sólo vamos a describir, además de lo visto en el apartado anterior, los sistemas basados en los estudios del grupo CODASYL.

En el lenguaje diseñado por CODASYL para la descripción, existen cuatro apartados distintos:

1.– Apartado general.

En él se determinan las características globales del esquema que se pretende describir.

Figura 124. Los principales fabricantes de ordenadores han diseñado sus propios softwares de bases de datos. Esta proliferación de productos puede conducir a la confusión de los usuarios. El grupo CODASYL ha intentado conseguir un lenguaje universal que pueda ser utilizado independientemente del ordenador disponible.

2.– Apartados de área.
En cada uno de los posibles apartados de área (pueden ser varios) se definen las distintas áreas en que pueden estar agrupados los registros.

3.– Apartados de registros.
En ellos se especifican las características de los registros que intervengan en la base. Fundamentalmente se determinan los campos integrantes del registro.

4.– Apartados de conjunto.
Se indican los agrupamientos de registros de distinto tipo en una clave especial de conjuntos CODASYL.

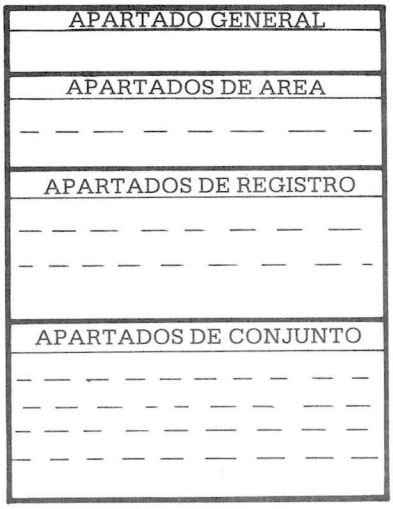


Figura 125. El lenguaje diseñado por CODASYL se basa en cuatro tipos de apartados. En el apartado general se especifica el esquema de la base de datos. En los apartados de área se indican las agrupaciones de los registros. En los apartados de registro se dan las características de los mismos. Y, por último, en los apartados de conjunto se ordenan los registros en conjuntos.

Para aclarar la estructura del lenguaje de datos CODASYL, vamos a suponer que la base de datos tiene tres tipos de registros: En uno se indican los datos correspondientes a los departamentos de la empresa; en otro los trabajos que se realizan en la empresa; y en el tercero se especifican los datos del personal de la empresa. Cada registro de tipo departamento puede estar conectado con varios registros de tipo trabajos (todos los trabajos que estén asignados al departamento) y con varios registros de tipo empleado (todos los empleados que estén integrados en el departamento).

Según esta descripción, la base se podría representar en forma de árbol de la siguiente manera:

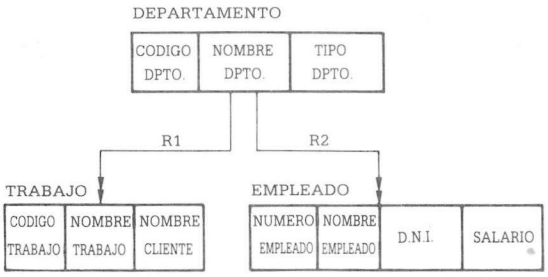


Figura 126.

En el anterior diagrama hemos determinado mediante R1 a la relación existente entre un departamento y los trabajos asignados a él, y mediante R2 a la relación existente entre un departamento y los empleados integrados en él.

La descripción de estos datos mediante el lenguaje CODASYL, se realizaría de la siguiente forma:

```
APARTADO GENERAL        ———→ SCHEMA NAME IS EMPRESA
APARTADO DE AREA        ———→ AREA NAME IS UNICA
APARTADO DE REGISTRO    ———→ RECORD NAME IS DEPARTAMENTO
                                 01 CODIGO-DEPTO
                                 01 NOMBRE-DEPTO
                                 01 TIPO-DEPTO
APARTADO DE REGISTRO    ———→ RECORD NAME IS TRABAJO
                                 01 CODIGO-TRABAJO
                                 01 NOMBRE-TRABAJO
                                 01 NOMBRE-CLIENTE
APARTADO DE REGISTRO    ———→ RECORD NAME IS EMPLEADO
                                 01 NUMERO-EMPLEADO
                                 01 NOMBRE-EMPLEADO
                                 01 D.N.I.
                                 01 SALARIO
APARTADO DE CONJUNTO    ———→ SET NAME IS R1
                                 OWNER IS DEPARTAMENTO
                                 MEMBER IS TRABAJO
                                 ASCENDING KEY IS CODIGO-
                                                  (TRABAJO
APARTADO DE CONJUNTO    ———→ SET NAME IS R2
                                 OWNER IS DEPARTAMENTO
                                 MEMBER IS EMPLEADO
                                 ASCENDING KEY IS NUMERO-
                                                  (EMPLEADO
```

3.1 Apartados del Lenguaje CODASYL

Como ya adelantábamos en el punto anterior, la descripción de los datos de la base se hace en cuatro tipos de apartados, los dos primeros no merecen ningún comentario puesto que sólo sirven para dar un nombre al esquema y al área.

En los apartados de registro se empieza dando el nombre del registro mediante la instrucción "RECORD NAME IS ...". A continuación se numeran todos los datos integrantes del registro. Esta nume-

ración no es ordinal, sino que determina el nivel ocupado dentro de la jerarquía del registro. Así, si un dato se numera con 01 se indica que simplemente pertenece al registro, en cambio si se numera con 02 se indica que este dato forma parte del dato de nivel 01 precedente. De esta forma, mediante la asignación de niveles, se pueden realizar unos subapartados cuando algún dato así lo requiera.

Alguno de los datos de nivel inferior (y por lo tanto pertenecientes a otro de nivel superior) pueden ser a su vez subdivididos en un nuevo nivel. La forma en que se realizan estas subdivisiones dentro del registro es muy similar a la utilizada tradicionalmente en el lenguaje COBOL.

Aunque en el ejemplo anterior sólo indicábamos para cada dato integrante de un registro su nivel y nombre, también es necesario especificar el formato que tendrá el dato, y esto se hace mediante la claúsula PICTURE, indicando por ejemplo si el dato es numérico o alfabético y el número de posiciones que ocupará (dígitos o caracteres).

El último apartado dedicado a los conjuntos CODASYL, implica un concepto completamente nuevo y propio de la base de datos. Un conjunto CODASYL consiste en un árbol de dos niveles. Un árbol puede formarse aunando varios conjuntos CODASYL de forma que se completen todos los niveles del árbol. Así pues, los analistas pueden definir estructuras arborescentes muy complejas, agregando en un único árbol varios conjuntos. De esta forma se puede llegar a relaciones complejas partiendo de otras más simples. La denominación de conjunto dada por CODASYL, no debe ser confundida con el concepto que se representa con el mismo nombre dentro de las Matemáticas. En el ámbito de las bases de datos jerárquicas se utiliza la denominación conjunto para designar una relación entre dos o más tipos de registros. Cada conjunto debe tener un tipo de registro declarado como propietario y uno o más tipos de registros de nivel inferior, declarados como miembros del propietario. En el ejemplo anterior teníamos que todo el árbol utilizado para representar la base de datos estaba constituido por un único conjunto.

El registro propietario era el denominado DEPARTAMENTO y sus miembros eran los tipos de registros denominados TRABAJO y EMPLEADO. Dado que cada registro DEPARTAMENTO puede estar relacionado con varios registros del tipo TRABAJO y con varios del tipo EMPLEADO, las relaciones determinadas por los conjuntos R1 y R2 son múltiples, como se indica al marcar con dos cabezas las flechas que conectaban a los registros.

Todos los apartados del esquema pueden ser modificados des-

pués de su diseño, sin que ello afecte al resto de los apartados.

3.2 Clasificación de los Conjuntos.

Todos los miembros de un conjunto CODASYL deben estar clasificados en un cierto orden que debe ser especificado en el diseño. El software de base será el encargado de que los registros miembros estén clasificados desde el punto de vista lógico según el orden indicado, aunque físicamente su ubicación pueda ser diferente. Cuando algún registro es miembro de más de un conjunto estará clasificado en cada uno de ellos según se haya indicado, sin que esto implique que el registro esté repetido en el almacenamiento físico.

Dentro del lenguaje CODASYL el diseñador de la base, además de indicar las características de clasificación lógica, puede decidir el ordenamiento físico. Aunque esto está especialmente reñido con la condición de independencia de los datos, ya que la modificación del almacenamiento físico puede implicar en este caso un cambio en algún subesquema y en el esquema. Precisamente por esto, no es recomendable la utilización de la posibilidad de que el diseñador elija un ordenamiento físico concreto.

Si bien la clasificación de los registros miembros de un conjunto puede y debe ser indicada, no está permitido especificar una clasificación para los registros propietarios. Aunque si un registro propietario es miembro de otro conjunto, sí podrá ser clasificado en esa condición.

Existen varias posibles clasificaciones para los registros miembros de un conjunto:

1.– Cuando debido a la utilización que se vaya a hacer de la base, la clasificación de los miembros de un conjunto sea indiferente, se deja que sea el software de base el que elija la ordenación que considere más conveniente. Esto se especifica mediante la instrucción ORDER IS INMATERIAL.

2.– Cuando se produce la inserción de nuevos miembros en un conjunto, existen cuatro posibilidades para indicar su clasificación:

   (1) Si se indica ORDER IS LAST, el nuevo miembro se ubicará al final de los ya existentes.

   (2) Si se indica ORDER IS FIRST, la colocación del nuevo registro se realiza al pricipio de los ya existentes.

   (3) Si se indica ORDER IS PRIOR, el nuevo miembro se sitúa inmediatamente antes de otro miembro determinado.

(4) Por último, si se indica ORDER IS NEXT, la colocación del nuevo miembro se realiza inmediatamente después de otro miembro determinado.

Un resumen de las características generales del conjunto CODASYL, que se describen exactamente en el Data Description Language Journal of Development, es el siguiente:

1.– Un tipo de conjunto es una relación entre tipos de registros.
2.– En un esquema se pueden declarar cualquier número de tipos de conjuntos.
3.– Cada tipo de conjunto debe tener un nombre y un tipo de registro propietario.
4.– Cada tipo de conjunto debe tener uno o más tipos de registros miembros.
5.– Cada tipo de conjunto debe tener una clasificación propia.
6.– Cualquier tipo de registro puede ser declarado en un esquema como propietario de uno o más conjuntos y simultáneamente, si se necesita, puede ser declarado como miembro de uno o más conjuntos.
7.– Ningún tipo de registro puede ser propietario y miembro de un mismo conjunto.
8.– La existencia del registro propietario es la que establece la existencia del conjunto.

3.3. Control y Selección.

Para preservar la independencia de los datos, se prohibe en el diseño del esquema hacer ninguna referencia a la organización física de los mismos. Algunos autores ven en esta concepción rigurosa un elemento negativo para el aprovechamiento óptimo de los recursos de almacenamiento. Evidentemente, este razonamiento puede ser válido en algunos casos, pero la optimización física debe ser una tarea realizada independientemente de la descripción lógica de los datos, consiguiendo por un lado mantener la independencia entre organización física y lógica y, por otro lado, optimizar el almacenamiento físico.

En el período de tiempo comprendido entre 1.960 y 1.970, la cuestión más cuidadosamente tratada en los centros de cálculo era la optimización del almacenamiento de datos. Esta cuestión resultó a la larga altamente negativa, ya que si bien los dispositivos de almacenamiento se explotaban de una forma acertada, los programas de aplicación dependían completamente de los datos, con lo que la más pequeña modificación de éstos implicaba la modificación de los pro-

gramas. En la actualidad, la tendencia es precisamente la contraria. Aun a riesgo de desaprovechar los dispositivos de almacenamiento, se cuida con mayor esmero la independencia entre datos y programas. No obstante, existen unos controles sobre la optimización del almacenamiento que conviene realizar:

1.– Cuando un grupo de datos se utilice simultáneamente, será conveniente que se almacenen los unos próximos a los otros, de forma que al realizar las operaciones de búsqueda, la paginación sea mínima.

2.– Los datos requeridos con mayor frecuencia deben ubicarse en las áreas de más fácil acceso; en cambio, aquellos cuya utilización sea esporádica pueden situarse en las áreas de más dificil acceso.

3.– Cuando los datos se almacenan en soportes de almacenamiento con cartuchos, los registros de acceso común deben disponerse en la misma celda, para así eliminar el tiempo perdido en cambiar de celda.

Por último, vamos a citar las posibilidades ofrecidas por el lenguaje CODASYL para la selección de conjuntos. Lo normal en una base de datos media, es que el número de conjuntos almacenados sea elevado. Surge así un nuevo problema: ¿cómo seleccionar dentro de la base de datos un conjunto determinado? y ¿dónde ubicarlo físicamente cuando se haya terminado su utilización?. Aunque el esquema no debe tener en cuenta estas cuestiones, puesto que son propias del almacenamiento físico, sí se puede especificar la forma general en que se usarán los conjuntos. Las posibilidades de utilización se pueden dividir en cuatro tipos:

1.– La selección del conjunto se realiza mediante un programa concreto. En este caso, en el esquema se indica el nombre del programa encargado de la selección.

2.– La selección del conjunto se realiza mediante el software de base. Para ello el programa de aplicación pasará el identificador del registro propietario del conjunto y el sistema le devolverá el conjunto solicitado.

3.– La selección del conjunto se consigue mediante un cálculo, realizado en función de una clave aportada por el programa de aplicación solicitante del conjunto.

4.– La selección del conjunto se realiza de forma transitiva, es decir, la selección se hace mediante un conjunto intermediario.

CUESTIONARIO CAPITULO 8

UTILIZACION DE BASES DE DATOS JERARQUICAS

1.– ¿Qué es un lenguaje de datos?
2.– ¿Qué características presenta el DL/I CODASYL?
3.– ¿Qué características tiene el lenguaje CODASYL?
4.– ¿Cómo está organizada una base de datos IMS?
5.– ¿Qué hace la Get Unique?
6.– ¿En qué se diferencia la operación GU de la GN?
7.– ¿Con qué instrucción se borran campos?
8.– ¿Qué apartados existen en el lenguaje CODASYL?
9.– ¿Qué clasificaciones son posibles para los registros miembros de un conjunto?
10.– ¿Qué controles de optimización de almacenamiento existen?

CAPITULO IX

UTILIZACION DE BASES DE DATOS EN RED

1.– SIMILITUD ENTRE LOS MODELOS JERARQUICO Y DE RED

En el lenguaje DDL de CODASYL, una condición fundamental era la existencia de un único padre para cada registro miembro de un conjunto, al que denominábamos registro propietario del conjunto. Sin embargo, cualquier registro miembro puede tener tantos padres como se necesite, siempre que pertenezcan a otros conjuntos. Luego el lenguaje DDL, puede ser utilizado tanto para definir la estructura de los datos de una base jerárquica, como para los de una base en red.

Esta compatibilidad es una consecuencia directa de la similitud entre los modelos jerárquicos y de red. Como ya hemos visto en capítulos anteriores, una red no es más que la generalización de un árbol y por lo tanto, en determinados casos se pueden utilizar operaciones muy similares para manejar ambas estructuras.

Si bien esta similitud permite utilizar el lenguaje DDL para la definición de redes con relaciones en las que haya correspondencia simple entre padres e hijos, no sirve cuando hay correspondencia compleja entre hijos y padres, aunque siempre es posible reducir la complejidad de la red (aumentando la redundancia) mediante subdivisiones hasta conseguir que pueda ser definida mediante el lenguaje DDL. Veamos un sencillo ejemplo de esquema con relaciones complejas intratables con DDL. Supongamos que tenemos varios proveedores y varios productos, de forma que cada proveedor puede su-

ministrar más de un producto y cada producto puede ser suministrado por más de un proveedor.

Tendríamos una red del siguiente tipo:

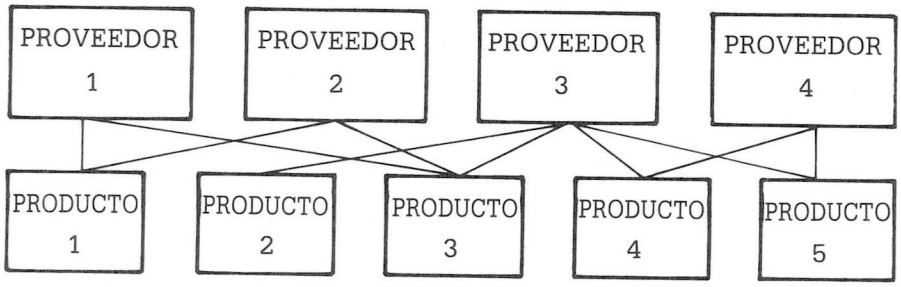


Figura 127.

Evidentemente, este esquema no puede ser definido mediante DDL, sin embargo, podría descomponerse de la siguiente manera:

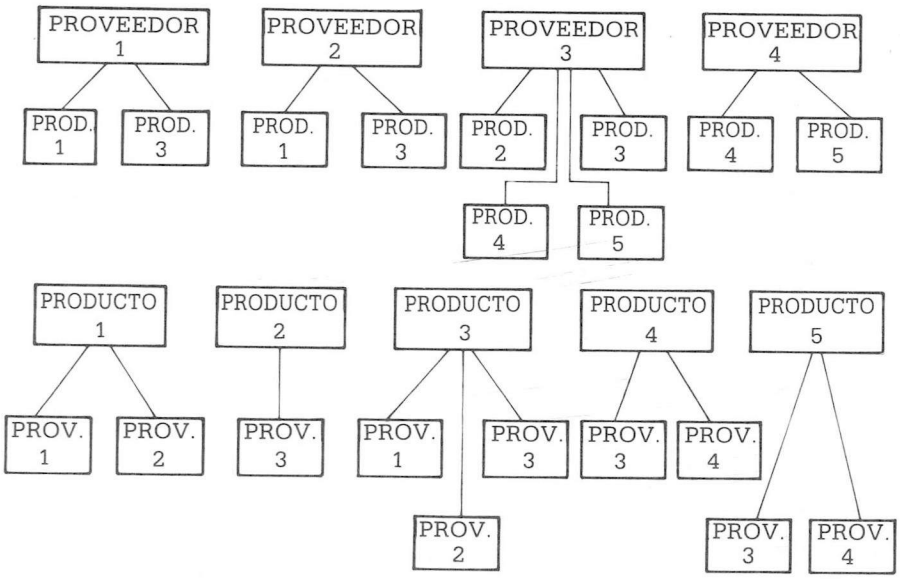


Figura 128.

cumpliendo así las condiciones necesarias para poder ser definida mediante DDL. Si el hecho de realizar esta simplificación estuviera motivado únicamente por la necesidad de adaptar la red al lenguaje

de CODASYL, habría que pensar que este lenguaje no es el apropiado para describir este modelo, pero ésto no es así. Como ya vimos, la simplificación de la red es útil desde cualquier punto de vista, y por lo tanto, el DDL se puede considerar como un buen lenguaje para la descripción de bases de datos en red.

Los dos principales problemas en el manejo de las redes son los ciclos y los lazos. Mediante el lenguaje de datos de CODASYL se pueden tratar perfectamente los ciclos, pero no es posible que un registro pueda ser a la vez miembro y propietario del mismo conjunto. Dicho de otra forma, el DDL no puede tratar lazos.

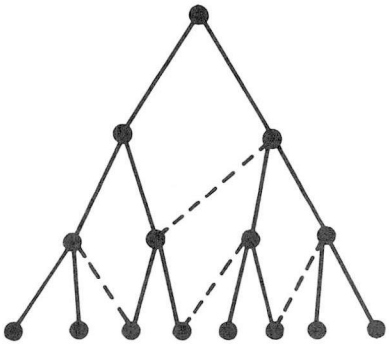


Figura 129. La estructura de red no es más que la generalización de una estructura arborescente. En la red de la figura están marcadas con trazos punteados las relaciones que estarían prohibidas en un árbol. Apoyándonos en esta propiedad, podemos afirmar que todo lenguaje que sirva para definir redes puede ser utilizado también para definir árboles pero no viceversa.

Dado que en el capítulo anterior hemos estudiado todas las características del lenguaje DDL y, que este puede ser utilizado cómodamente para describir esquemas de redes, no es necesario que nos extendamos más en este apartado.

2.– LENGUAJES PARA BASES DE DATOS EN RED

El resultado del grupo DBGT-CODASYL en su estudio sobre bases de datos jerárquicas y de red, apuntaba a la necesidad de utilizar tres lenguajes para la definición y consulta sobre cualquier base de datos:

1.– Para definir el esquema de la base de datos en red se puede utilizar el lenguaje DATA DEFINITION LANGUAJE (DDL).

2.– Para definir los distintos subesquemas de la base también se puede utilizar el lenguaje DDL aunque en distinto entorno.

3.– Por último, para manipular los datos desde programas de aplicación, existe un lenguaje especialmente diseñado para el manejo de redes que se denomina DATA MANIPULATION LANGUAJE (DML).

De esta forma, el manejo de una base de datos en red se puede representar mediante la siguiente figura:

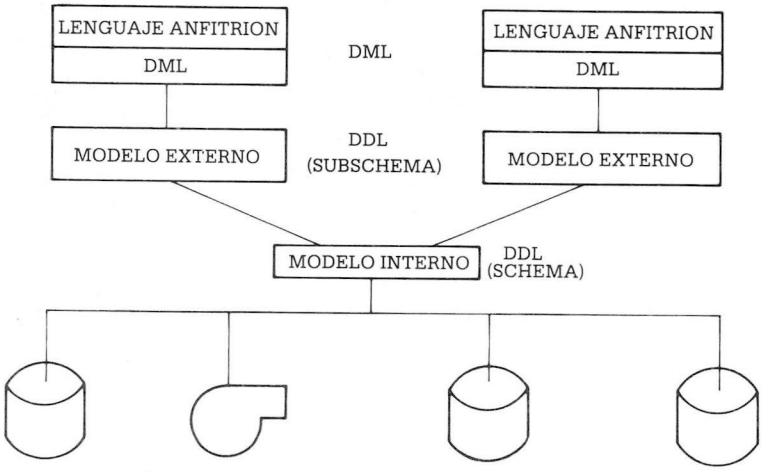


Figura 130.

La orientación del DDL, tanto en entorno SCHEMA como SUBSCHEMA, es similar a la del lenguaje de programación COBOL; en cambio el DML tiene unas características particulares.

2.1. Consideraciones Prácticas.

Para aclarar conceptos sobre las bases de datos en red vamos a describir un modelo como ejemplo:

Supongamos una empresa formada por tres departamentos D_1, D_2 y D_3, y cuatro empleados E_1, E_2, E_3 y E_4. Los empleados E_1, E_2 y E_3 están encuadrados en el departamento D_1 y el empleado E_4 pertenece al departamento D_2. El departamento D_3 no tiene ningún empleado asignado. Tenemos pues:

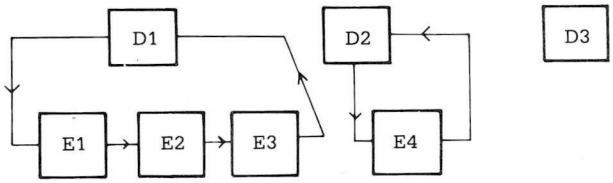


Figura 131.

Dos conjuntos con elementos que sirven para representar la ligazon que une a los departamentos con los empleados, y un conjunto que sirve para representar la ausencia de miembros para el propietario D_3.

Si suponemos que los departamentos D_1 y D_2 están incluidos en el área empresarial A_1 y el departamento D_3 en el área A_2, se establecería un nuevo conjunto con lo que la estructura de conjuntos de la red sería la siguiente:

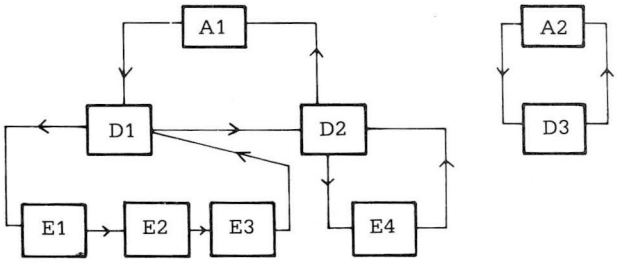


Figura 132.

Como ya habíamos visto al estudiar las posibilidades del lenguaje DDL, los enlaces utilizados para determinar los conjuntos pueden ser portadores de alguna información adicional.

2.2. Lenguaje de Consulta DML.

Este lenguaje de consulta para bases de datos en red, consiste en una serie de operaciones que pueden ser utilizadas desde el lenguaje anfitrión para acceder al dato deseado. Las operaciones fundamentales del DML son las siguientes:

1.– FIND.

Se encarga de localizar un registro concreto de la base de datos, mediante la imposición de una serie de condiciones que

deben ser satisfechas por el registro a localizar. Esta operación sólo sirve para encontrar un registro determinado pero no lo deja a disposición del programa.

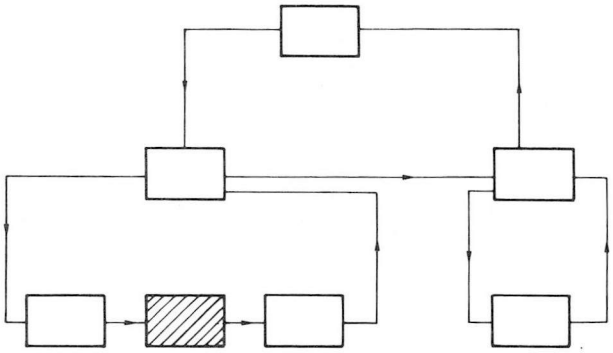


Figura 133. La operación FIND busca un registro dentro de la red a partir de unas condiciones especificadas. Su misión consiste única y exclusivamente en completar la búsqueda devolviendo el registro localizado o una señal que indique la no existencia de tal registro.

2.– GET.

Permite que el programa de aplicación disponga de un registro ya localizado. Por lo tanto, sólo se puede ejecutar una operación GET después de haber ejecutado una operación FIND. Después de realizarse la GET, el programa puede utilizar libremente la información contenida en el registro.

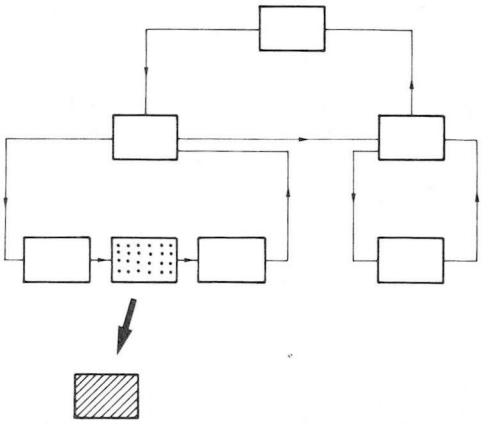


Figura 134. Ejecutando una operación GET se transfiere el registro localizado previamente, mediante una operación FIND, a la memoria principal para que el programa anfitrión pueda realizar sobre él los procesos oportunos.

3.– MODIFY.

Mediante esta operación se pueden modificar los campos de la unidad que esté activa.

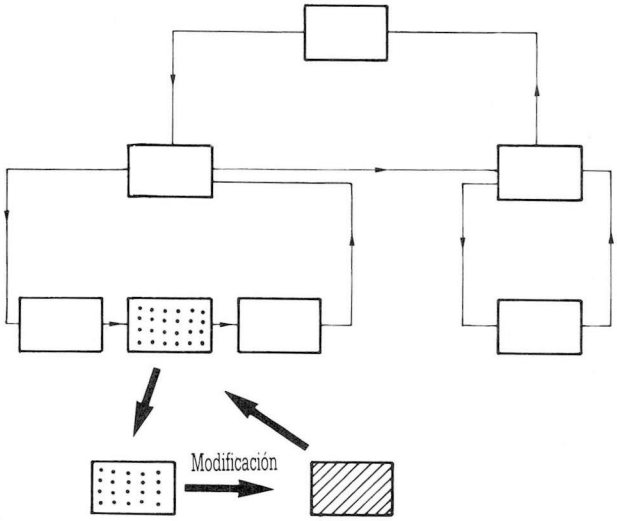


Figura 135. La operación MODIFY permite reemplazar los datos de un registro localizado y accedido; para ello devuelve a la red el registro situado en la memoria principal.

4.– CONNECT.

Se utiliza para conectar la unidad activa como nuevo elemento de un conjunto localizado previamente.

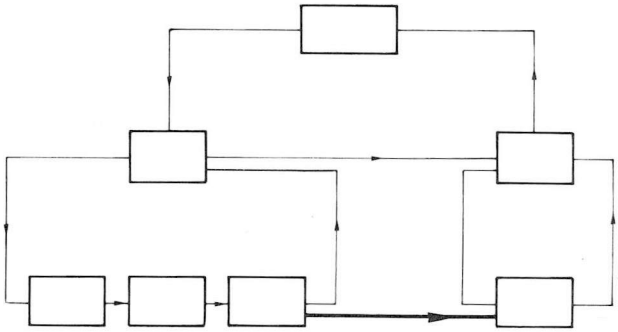


Figura 136. Mediante la operación CONNECT se pueden conectar dos registros ya existentes en la red. En la figura se ha marcado con una flecha en negrita una nueva conexión establecida mediante una operación CONNECT.

5.– DISCONNECT.

Sirve para lo contrario que la operación anterior, es decir, elimina la conexión entre la unidad activa y un conjunto localizado previamente.

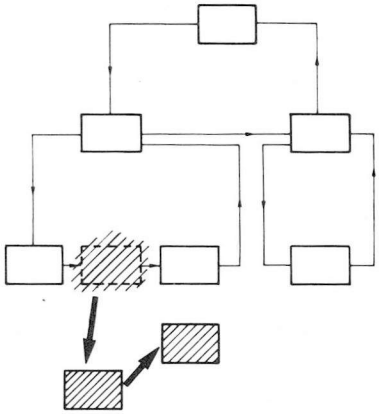


Figura 137. La operación DISCONNECT sirve para desconectar un registro de la red, dejándolo, no obstante, almacenado en la base de datos. Si dicho registro sólo estaba conectado en un conjunto, el registro quedará aislado. Pero si pertenecía a otros conjuntos, seguirá incluído en ellos.

6.– ERASE.

Cuando se ejecuta una operación Erase se elimina totalmente la unidad activa.

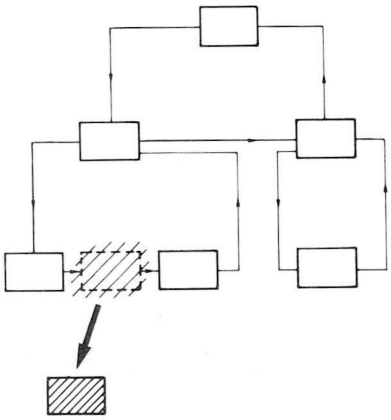


Figura 138. La operación ERASE sirve para eliminar totalmente un registro de la base de datos. Por supuesto, dicho registro debe haber sido localizado previamente.

7.– STORE.

Al contrario que la operación anterior, al ejecutar una operación Store se introduce la unidad activa en la base de datos.

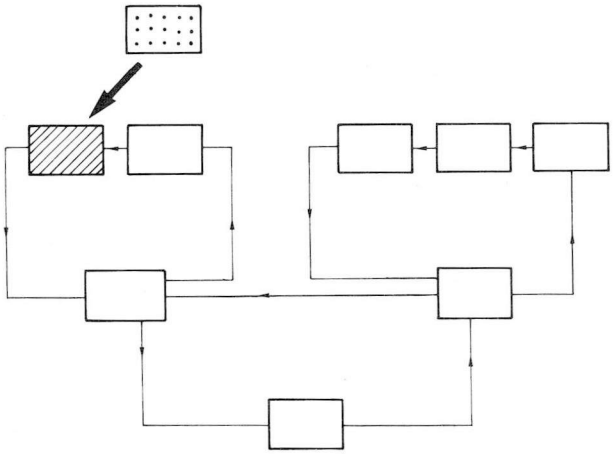


Figura 139. Mediante la operación STORE se puede incluir un registro en la base de datos, pero es necesario cargar sus datos en memoria principal e indicar en qué situación debe ser incluído.

CUESTIONARIO CAPITULO 9

UTILIZACION DE LAS BASES DE DATOS EN RED

1.– ¿Cuáles son los principales problemas en el manejo de las redes?
2.– ¿Qué orientación tiene el DDL?
3.– ¿Qué tipo de lenguaje es el DML?
4.– ¿Qué operación es la FIND?
5.– ¿Qué permite la operación GET?
6.– ¿Qué realiza la operación MODIFY?
7.– ¿Qué diferencias hay entre las operaciones CONNECT y DISCONNECT?
8.– ¿Qué ejecuta el ERASE?

CAPITULO X

UTILIZACION DE BASES DE DATOS RELACIONALES

1.– CONCEPTOS GENERALES

Las bases de datos relacionales son sin duda las que tienen un fundamento teórico más abundante. Precisamente por esto se puede realizar una descripción más precisa que en los modelos anteriores, utilizando una notación más matemática y por lo tanto más precisa. Si se comparan los índices de los tres capítulos dedicados a los tres modelos estudiados, se verá claramente que la estructuración de las materias es mucho más concreta en el caso de las bases relacionales.

La mayoría de los autores están de acuerdo en afirmar que el modelo relacional es el más adecuado para conseguir los objetivos perseguidos. No obstante también es el que más recursos consume. Para que este modelo acabe de imponerse definitivamente, sería necesaria la aparición de equipos más potentes que los actuales, capaces de manejar eficientemente la complicada estructura física con que se almacenan los datos.

Como comprobaremos en este capítulo, las bases de datos relacionales son las más sencillas de utilizar. Para conseguir esta sencillez es necesario que el software de base sea potente y complejo (no de manejo pero sí en consumo de CPU). En resumen, en la actualidad las bases de datos relacionales son en teoría superiores a las jerárquicas y de red, aunque en la práctica cuando se tiene que tratar un gran volumen de datos no es tan clara la superioridad, debido a la cantidad de recursos consumidos.

En este capítulo comenzaremos dando una definición formal de los principales conceptos manejados, aplicándolos en un sencillo ejemplo. A continuación pasaremos a estudiar un concepto fundamental, como es la dependencia funcional entre atributos. Después veremos cómo se puede ir mejorando el funcionamiento de una base de datos relacionales mediante un proceso de normalización y aplicaremos dicho proceso al mismo ejemplo de base de datos utilizado en las definiciones. Para terminar estudiaremos las principales instrucciones de dos de los lenguajes más utilizados para trabajar con bases de datos relacionales: el lenguaje ALFA y el lenguaje QUERY BY EXAMPLE.

2.– DEFINICIONES

A continuación se presentan algunos de los conceptos y definiciones más importantes subyacentes en el modelo relacional. La mayor parte de estos conceptos fueron introducidos por Cood en 1.970.

Relación.

Dados n conjuntos D_1, D_2, ...D_n llamados dominios (no necesariamente distintos), una relación R sobre ellos es un subconjunto del producto cartesiano $D_1 \times D_2 \times ...D_n$.

Grado de la Relación.

Se llama grado de la relación, al número n de conjuntos en que se basa.

Dominio.

Nos referiremos a D_i como el i-ésimo dominio de la relación.

N-Tupla.

Se llama n-tupla o simplemente tupla, a cada uno de los elementos de una relación de grado n.

Cardinalidad de la Relación.

Se llama cardinalidad de R, al número de n-tuplas que contiene.

Atributo.

Dado que los dominios de una relación no son necesariamente disjuntos, llamaremos atributos a cada uso particular de un dominio.

Base de Datos Relacional.

Una base de datos relacional R es un conjunto finito de relaciones variables con el tiempo, definidas sobre un conjunto finito de dominios D_1, D_2, ... D_n.

Modelo.

En un sistema de gestión de base de datos relacional integrado, es usual que los distintos usuarios deban utilizar distintos subconjuntos del universo total de datos. Se llama modelo al universo total de datos, es decir, al conjunto completo de todas las relaciones almacenadas en la base de datos.

Esquema.

Esquema es el conjunto de declaraciones que define el modelo.

Submodelo.

Entendemos por submodelo al conjunto de relaciones a las que puede acceder un usuario determinado.

Subesquema.

Se llama subesquema al conjunto de declaraciones que definen un submodelo.

Sistema de Gestión.

Un sistema de gestión de base de datos BD completo debe suministrar medios para la definición del esquema y subesquema a cualquier tipo de usuario.

Tabla.

Aunque no es esencial, se acostumbra a representar cada relación mediante una tabla, en la que cada fila representa una n-tupla.

Orden de los Elementos.

En principio no existe ningún orden en los elementos de una relación R, pero existe un orden implícito establecido por la secuencia seguida en la carga de los datos de cada relación.

Ejemplo.

Para ilustrar las anteriores definiciones, vamos a realizar una re-

presentación tabular de las relaciones correspondientes a una BD de una empresa, que contiene datos sobre proveedores, componentes y pedidos. La descripción de las relaciones es:

1.– PROVEEDORES

P	Identificador del proveedor.
NOM-P	Nombre del proveedor.
CIUDAD	Ciudad de residencia del proveedor.
REGION	Región de residencia del proveedor.
DIR	Dirección del proveedor.

2.– BENEFICIOS

NOM-P	Nombre del proveedor.
BENEF	Beneficio obtenido.
TIPO	Tipo de proveedor.
AÑO	Año.

3.– COMPONENTES

C	Identificador del componente.
NOM-C	Nombre del componente.
COLOR	Color del componente.

4.– PEDIDOS

P	Identificador del proveedor de un pedido.
C	Identificador del componente de un pedido.
CANT	Cantidad total pedida.
P-M	Precio por metro.

5.– DIRECCIONES

CIUDAD	Ciudad.
CALLE	Calle.
COD-P	Código postal.

Supondremos que independientemente de las características semánticas que son inherentes a los propios datos, se verifican:

1.– Los valores de COD-P son números de cuatro cifras, en los que el primer par de cifras representa la información sobre la ciudad de referencia y las restantes sobre el distrito postal en la ciudad.
2.– Algunos proveedores pueden serlo de varios componentes distintos.
3.– Los valores del dominio BENEF son totales y por consiguiente únicos por proveedor y año.
4.– Todos los componentes tienen el mismo precio por metro en todos los proveedores.
5.– El orden de las filas de cada relación no es significativo.

6.– No puede haber dos filas iguales en ninguna relación.

Supongamos que el contenido de la base de datos en un primer momento es:

PROVEEDORES

P #	NOM-PROV.	CIUDAD	REGION	DIRECCION
1	RUIZ	LUGO	GALICIA	MAR, 13
2	LOPEZ	SORIA	CASTILLA	SOL, 22
3	SAEZ	SORIA	CASTILLA	CIES, 7
4	GIL	LUGO	GALICIA	ROSA, 2
5	ROY	LEON	LEON	REY, 88

BENEFICIOS

NOM.-PROV.	BENEFICIOS	TIPO	AÑO
ROY	20	TELAS	1975
ROY	30	TELAS	1976
RUIZ	10	TELAS	1975
RUIZ	15	TELAS	1976
RUIZ	10	FORROS	1975
RUIZ	15	FORROS	1976
SAEZ	3	TELAS	1975
SAEZ	5	TELAS	1976

COMPONENTES

C #	NOM.-C	COLOR
1	TERGAL	AZUL
2	LANA	NEGRO
3	NYLON	ROJO
4	NYLON	AZUL
5	PANA	ROJO
6	FORRO	AZUL

DIRECCIONES

CIUDAD	CALLE	COD.-P.
LUGO	MAR	4325
LUGO	REY	4336
SORIA	SOL	6512
SORIA	CIES	6518
LEON	REY	4413

PEDIDOS

P #	DATOS DEL PEDIDO		
	C #	CANTIDAD	P-M
1	1	3	120
	2	2	170
	3	4	170
	4	2	140
	5	1	120
	6	1	150
2	1	3	120
	2	4	170
3	3	4	170
	5	2	120
4	2	2	170
	4	3	140
	5	4	120
5	5	5	120

Figura 140.

A continuación veremos un ejemplo de cada una de las definiciones originales:

– RELACION

COMPONENTES

C #	NOM.-COM.	COLOR
1	TERGAL	AZUL
2	LANA	NEGRO
3	NYLON	ROJO
4	NYLON	AZUL
5	PANA	ROJO
6	FORRO	AZUL

Figura 141.

– GRADO DE LA RELACION COMPONENTES
3

– DOMINIO
COLOR = {AZUL, NEGRO, ROJO} o
COLOR = CADENA DE HASTA 10 CARACTERES

– 3-TUPLA

2	LANA	NEGRO

– CARDINALIDAD DE LA RELACION
Card (COMPONENTES) = 6

– ATRIBUTO
COLOR: en este caso el dominio color se utiliza únicamente en una relación, pero podría ser también el dominio de un atributo de otra relación.

– BASE DE DATOS RELACIONAL
Conjunto de todas las relaciones.

– MODELO
{PROVEEDORES, BENEFICIOS, COMPONENTES, DIRECCIONES, PEDIDOS}

– ESQUEMA
RELACIONES: PROVEEDORES, ...
DOMINIOS: NOM-P, ...
.
.
.

– SUBMODELO
BENEFICIOS, PROVEEDORES, DIRECCIONES

– SUBESQUEMA
RELACIONES: .....
DOMINIOS: .....

.
.
.

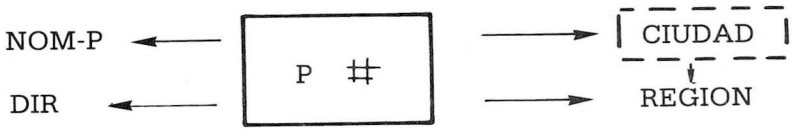
3.– DEPENDENCIA FUNCIONAL ENTRE ATRIBUTOS

3.1. Dependencia Funcional (DF).

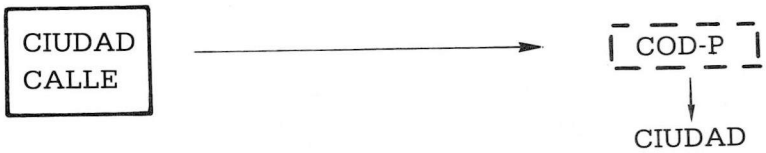
Sean dos colecciones de atributos A y B de una relación R. Se dice que B es funcionalmente dependiente de A y se nota RA ⟶ RB, sólo si en cada instante cada valor de A en R está asociado con un solo valor de B. En todos los casos en que no hay posible confusión, y con el fin de simplificar la notación, en lugar de RA ⟶ RB escribiremos A ⟶ B.

Ejemplos:

1.– En la relación PROVEEDORES existen las siguientes dependencias funcionales:



2.– En la relación DIRECCIONES:



3.2. Dependencias Funcionales Triviales.

Sea R una relación, las dependencias funcionales triviales son del tipo:

$$X \longrightarrow \emptyset \qquad o \qquad X \longrightarrow X$$

3.3. Propiedades de las Dependencias Funcionales.

• Reflexividad. Si Y está incluido en X ($Y \subset X$), entonces $X \longrightarrow Y$.

Esta propiedad es obvia, pues en cada instante cada valor de Y tiene asociado a los sumo uno en X (el mismo valor).

- Aumentación: Si $X \longrightarrow Y$ y $W \subset Z$, entonces $X \cup Z \longrightarrow Y \cup W$. Esta propiedad es evidente por la definición de dependencia funcional y por la anterior propiedad.
- Pseudotransitividad: Si $X \longrightarrow Y$ e $Y \cup Z \longrightarrow W$, entonces $X \cup Z \longrightarrow W$. También evidente por definición.
- Unión: Si $X \longrightarrow Y$ y $X \longrightarrow Z$, entonces $X \longrightarrow Y \cup Z$. En efecto, aplicando la aumentación a $X \longrightarrow Z$, se tiene que $X \cup Y \longrightarrow Z \cup Y$ (ya que $Y \subset Y$). Tenemos pues: $X \longrightarrow Y$ y $X \cup Y \longrightarrow Z \cup Y$ y por pseudotransitividad $X \longrightarrow Z \cup Y$.
- Descomposición: Si $X \longrightarrow Y$ y $Z \subset Y$, entonces $X \longrightarrow Z$. En efecto, por la reflexivilidad $Y \longrightarrow Z$ y como $X \longrightarrow Y$, por la pseudotransitividad $X \longrightarrow Z$.
- Proposición: $X \longrightarrow A \cup B \cup \ldots \cup K$ es equivalente a $X \longrightarrow A$, $X \longrightarrow B$, ... $X \longrightarrow K$. La demostración es inmediata a partir de las dos últimas propiedades.

3.4. Cierre de un Conjunto de DF.

Al conjunto de DF que se obtiene por sucesivas aplicaciones de las propiedades anteriores a un conjunto F inicial de DF, se le denomina cierre de F y se nota F_+ .

3.5. Cierre de un Conjunto de Atributos.

Dado un conjunto de atributos X, se define el cierre de X (relativo a F) y se nota X_+ , como el conjunto de todos los atributos A, tales que $X \longrightarrow A$ está en F_+ .

3.6. Cobertura.

Dado un conjunto F de DF, se dice que un conjunto H es una cobertura de F, si tiene el mismo cierre que F, es decir, si $H_+ = F_+$.

3.7. Cobertura no Redundante.

Una cobertura se dice que es no redundante si no contiene ningún subconjunto que a su vez sea cobertura.

3.8. Dependencia Redundante.

Una dependencia funcional $f \in F$ se dice redundante si pertenece al cierre de $F - \{f\}$.

3.9. Cobertura Redundante.

Una cobertura se dice redundante si, y sólo si, contiene dependencias funcionales redundantes.

3.10. Superclave.

Se dice que K es una superclave si es un conjunto de atributos de una relación del que dependen funcionalmente los restantes.

3.11. Clave.

Se dice que una superclave es una clave si es una superclave mínima, es decir, si no existe otra superclave que esté contenida en ella.

3.12. Atributo Primario.

Se dice que un atributo es primario en una relación si sólo aparece en la menor clave.

3.13. Dependencia Funcional Total.

Sean D y E dos colecciones distintas de atributos de R, siendo E funcionalmente dependiente de D. Se dice que E depende funcionalmente de forma total de D, sólo sí E no es dependiente funcionalmente de ningún subconjunto de D distinto del total.

3.14. Dependencia Transitiva.

Sean A, B y C tres colecciones distintas de atributos en una relación R cuyo grado es mayor o igual a tres. Supongamos que se verifican tres condiciones:

$A \longrightarrow B$, $B \not\longrightarrow A$ y $B \longrightarrow C$.

Se dice que C es transitivamente dependiente de A si $A \longrightarrow C$ y $C \not\longrightarrow A$.

4.– NORMALIZACION

El concepto de normalización fué introducido por Cood en 1.970. El motivo de la aparición de este concepto fué la observación de que para idénticos datos, diferentes relaciones entre los mismos presentaban muy distintas propiedades en cuanto a actualización, inserción, etc.

Vamos a describir algunas de las anomalías que pueden presentarse en una base de datos relacional:

4.1. Anomalías en las Actualizaciones.

Consideremos la relación PEDIDOS. Si al actualizar la tupla ⟨5,5,5,120⟩ observamos un error correspondiente al P-M (el precio del componente 5 ha sido modificado), y cambiamos en esta tupla 120 por 150, es evidente que la relación total quedaría en inconsistencia con respecto al precio de la componente 5. Por otro lado, el cambio de 120 por 150 en toda la relación, introduciría un error en las tuplas correspondientes a C #=1. Se hace pues necesaria una lectura de toda la relación con el fin de que los cambios se realicen sólo en las tuplas adecuadas. Si la relación anterior se hubiera descompuesto inicialmente en dos relaciones distintas con dominios respectivos (P #, C #, CANT) y (C #, P-M) las dificultades antes descritas no habrían aparecido.

4.2. Anomalías en las Inserciones.

Supongamos que en la relación PROVEEDORES queremos introducir dos nuevas tuplas, de las que sólo conocemos NOM-P, CIUDAD y REGION. El modo de hacerlo sería introduciendo dos tuplas con valores nulos en el dominio desconocido, lo cual sería imposible en el caso de existir una clave definida en el dominio P#.

4.3. Anomalías en las Eliminaciones.

Supongamos que en la relación PROVEEDORES eliminamos al proveedor de nombre "ROY", lo que equivaldría a eliminar en esta relación la última tupla. En este caso perderíamos la información de que la ciudad de "LEON" está en la provincia de "LEON", cosa que no ocurriría si desglosasemos esta relación en dos relaciones distintas: una para proveedores y otra para zonas.

La teoría que vamos a describir se basa en una serie de formas normales (primera, segunda, tercera, ...) que proporcionan sucesivas mejoras en cuanto a las propiedades de modificación de una base de datos.

4.4. Primera Forma Normal.

Se dice que R es una primera forma normal, solamente si ninguna de sus tuplas tiene elementos que a su vez sean conjuntos.

Se dirá que una relación es no normalizada si no es una primera forma normal.

Con respecto a las relaciones de la base de datos del ejemplo anterior, todas son primera forma normal excepto la relación PEDIDOS,

que es no normalizada como se puede observar en la tupla:

2	1 2	3 4	120 170

El paso de esta relación a una en primera forma normal, se obtiene inmediatamente reescribiendo la relación en la siguiente forma:

PEDIDOS-1

P #	C #	CANTIDAD	P-M
1	1	3	120
1	2	2	170
1	3	4	170
1	4	2	140
1	5	1	120
1	6	1	190
2	1	3	120
2	2	4	170
3	3	4	170
3	5	2	120
4	2	2	170
4	4	3	140
4	5	4	140
5	5	5	120

Figura 142.

El conjunto de relaciones B = {(PROVEEDORES, BENEFICIOS, COMPONENTES, DIRECCIONES, PEDIDOS} constituían una base de datos no normalizada, pero B_1 = {PROVEEDORES, BENEFICIOS, COMPONENTES, DIRECCIONES y PEDIDOS-1} forman una base de datos relacional en primera forma normal.

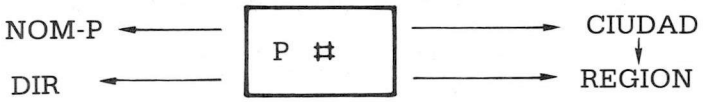
4.5. Segunda Forma Normal.

Se dice que una relación R es segunda forma normal si:

1.– Es una primera forma normal.
2.– Todo atributo que no sea primario es totalmente dependiente de cada clave posible de R.

Todas las relaciones de B_1 cumplen la primera condición de forma trivial. Veamos la segunda:

PROVEEDORES
– CLAVES POSIBLES:

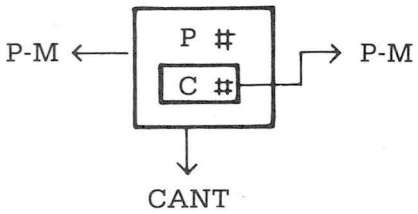


Como todos los atributos de la relación sólo dependen funcionalmente de P#, la única clave posible es P#.

Todos los atributos no primarios dependen totalmente de P#, luego se verifica la segunda condición y por lo tanto PROVEEDORES es una segunda forma normal.

Análogamente se puede comprobar que las relaciones BENEFICIOS, COMPONENTES y DIRECCIONES son segundas formas normales cuyas respectivas claves son: NOM-P, AÑO, C# y CIUDAD, CALLE.

En cambio la relación PEDIDOS-1 no es segunda forma normal, puesto que su clave es P#, C# y, en efecto, todos los atributos no primarios dependen funcionalmente de la clave, pero esta dependencia no es de forma total ya que el atributo P-M depende funcionalmente también sólo de C#.



Una manera de convertir la base de datos B_1 en segunda forma normal, es eliminando el dominio P-M de la relación PEDIDOS-1 y agregando uno equivalente en la relación componentes, con lo que tendríamos:

COMPONENTES-1

C #	NOMBRE-C	COLOR	P-M
1	TERGAL	AZUL	120
2	LANA	NEGRO	170
3	NYLON	ROJO	170
4	NYLON	AZUL	140
5	PANA	ROJO	120
6	FORRO	AZUL	190

Figura 143.

PEDIDOS-2

P #	C #	CANTIDAD
1	1	3
1	2	2
1	3	4
1	4	2
1	5	1
1	6	1
2	1	3
2	2	4
3	3	4
3	5	2
4	2	2
4	4	3
4	5	4
5	5	5

Figura 144.

y así podríamos formar:

B_2 = {PROVEEDORES, BENEFICIOS, COMPONENTES-1, DIRECCIONES, PEDIDOS-2} que ya está en segunda forma normal.

4.6. Tercera Forma Normal.

Decimos que una relación R está en tercera forma normal si:

1.– Es una segunda forma normal.

2.– Cada atributo que no sea primario, no es transitivamente dependiente de cada clave posible de R.

Todas las relaciones de B_2 cumplen la primera condición, pero la relación PROVEEDORES no la cumple, ya que:

P# $\longrightarrow$ CIUDAD y CIUDAD $\not\longrightarrow$ P#,
CIUDAD $\longrightarrow$ REGION y REGION $\not\longrightarrow$ CIUDAD, y
P# $\longrightarrow$ REGION y REGION $\not\longrightarrow$ CIUDAD,

por lo tanto, REGION es transitivamente dependiente de P#.

La forma de pasar a tercera forma normal es descomponer la relación PROVEEDORES en las relaciones PROVEEDORES-1 y ZONA, con lo que habremos roto la dependencia transitiva:

PROVEEDORES-1

P #	NOMBRE-P	CIUDAD	DIRECCION
1	RUIZ	LUGO	MAR, 13
2	LOPEZ	SORIA	SOL, 22
3	SAEZ	SORIA	CIES, 7
4	GIL	LUGO	ROMA, 2
5	ROY	LEON	REY, 88

ZONA

CIUDAD	REGION
LUGO	GALICIA
SORIA	CASTILLA
LEON	LEON

Figura 145.

con lo cual B_3 = {PROVEEDORES-1, ZONA, BENEFICIOS, COMPONENTES-1, DIRECCIONES, PEDIDOS-2}es una base relacional en tercera forma normal.

4.7. Otras Formas Normales.

Una vez que hemos conseguido que una base de datos esté en tercera forma normal, es de suponer que su funcionamiento es razonablemente bueno. No obstante existen otras formas normales como la de BOYCE-CODD, la cuarta forma normal y la quinta forma normal.

A continuación detallaremos someramente las características y mejoras de estos nuevos procesos de normalización.

• *Forma normal de Boyce-Codd.*

Una relación está en forma normal Boyce-Codd si:

1.– Está en tercera forma normal.

2.– Para todo par de conjuntos de atributos X, Y tales que $X \neq \emptyset$ $Y \neq \emptyset$ $X \cap Y = \emptyset$ si $X \longrightarrow Y$, entonces X es una superclave de R. Con otras palabras, si algún atributo depende de X, todos los demás también.

Todas las relaciones de B_3 son forma normal de Boyce-Codd ex-

cepto la relación DIRECCIONES, ya que:

$\boxed{\text{CIUDAD, CALLE}} \longrightarrow$ COD/P

$\boxed{\text{COD/P}} \longrightarrow$ CIUDAD

y por lo tanto, X = {CIUDAD, CALLE}, Y = {COD.P}, verificándose que $X \neq \emptyset$ $Y \neq \emptyset$ $X \cap Y = ç$, $X \longrightarrow Y$ sin ser X una superclave.

El paso de B_3 a una base en forma normal de Boyce-Codd se consigue sin más que descomponer la relación DIRECCIONES en dos:

DIRECCIONES-A

CALLE	COD.P.
MAR	25
SOL	36
CIES	12
ROMA	18
REY	19

DIRECCIONES-B

CIUDAD	COD.-C.
LUGO	43
SORIA	65
LEON	36

Figura 146.

y $B_{B\text{-}C}$ = {PROVEEDORES-1, ZONA, BENEFICIOS, COMPONENTES-1, DIRECCIONES-A, DIRECCIONES-B, PEDIDOS-2 }.

4.8. Cuarta Forma Normal.

La mejora introducida por la cuarta forma normal sobre la forma normal de Boyce-Codd es la eliminación de redundancia.

Por ejemplo, en la relación BENEFICIOS se puede hacer una descomposición de la siguiente manera:

BENEFICIOS-A

NOM-P	TIPO
ROY	TELAS
RUIZ	TELAS
RUIZ	FORROS
SAEZ	TELAS

BENEFICIOS-B

NOM-P	BENEFICIO	AÑO
ROY	20	1.975
ROY	30	1.976
RUIZ	10	1.975
RUIZ	15	1.976
SAEZ	3	1.975
SAEZ	5	1.976

Figura 147.

con lo que B_4 = {PROVEEDORES-1, ZONA, BENEFICIOS-A, BENEFICIOS-B, COMPONENTES-1, DIRECCIONES-A, DIRECCIONES-B, PEDIDOS-2} está en cuarta forma normal.

4.9. Quinta Forma Normal.

Se dice que R es quinta forma normal sí es cuarta forma normal y los JOIN sólo pueden realizarse mediante los atributos claves.

Un JOIN es la combinación de dos relaciones con un dominio común, que se puede denominar también unión de relaciones.

Para terminar este apartado vamos a resumir todo el proceso de normalización descrito, indicando las facciones que se deben tomar para progresar en la normalización, en el siguiente esquema de niveles de normalización:

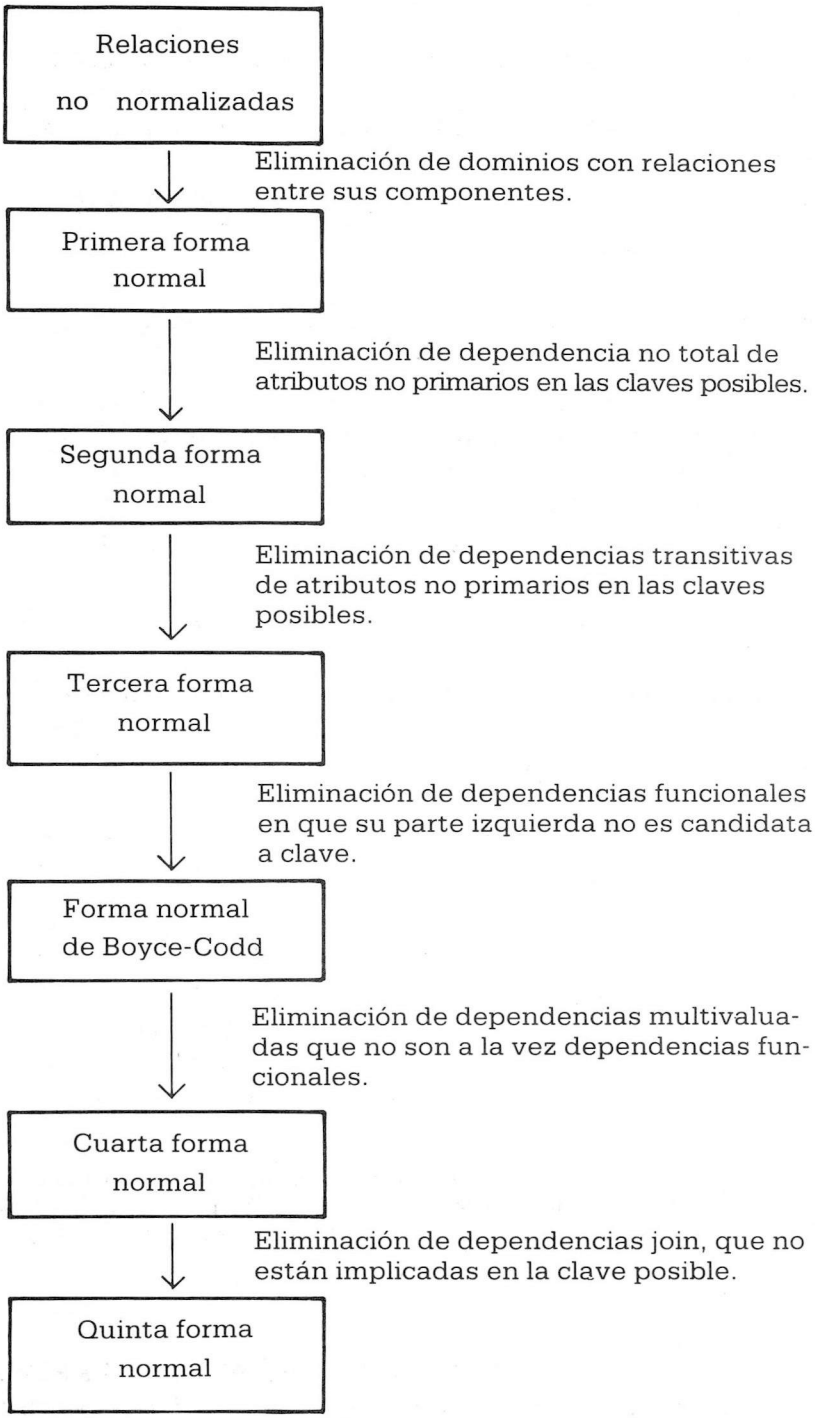

Relaciones no normalizadas
Eliminación de dominios con relaciones entre sus componentes.
Primera forma normal
Eliminación de dependencia no total de atributos no primarios en las claves posibles.
Segunda forma normal
Eliminación de dependencias transitivas de atributos no primarios en las claves posibles.
Tercera forma normal
Eliminación de dependencias funcionales en que su parte izquierda no es candidata a clave.
Forma normal de Boyce-Codd
Eliminación de dependencias multivaluadas que no son a la vez dependencias funcionales.
Cuarta forma normal
Eliminación de dependencias join, que no están implicadas en la clave posible.
Quinta forma normal

Figura 148.

5.– LENGUAJE ALFA

El lenguaje ALFA para la consulta a una base de datos relacional, está basado en el cálculo de predicados. Por ello comenzaremos por describir las características generales del cálculo de predicados.

La estructura general de llamada es:

COMANDO ZONA-MEMORIA (SOLICITUD)
RESTRICCIONES

Donde:

COMANDO debe ser alguna de las palabras reservadas del lenguaje ALFA.

ZONA-MEMORIA representa el nombre externo de las posiciones de memoria en que se almacenará la información. Este nombre externo deberá seguir las reglas del lenguaje de programación desde el que se invoque al comando.

SOLICITUD es la lista de atributos de una o más relaciones que se desean como respuesta.

RESTRICCIONES son las condiciones que deben ser verificadas para la inclusión o no en el resultado.

Dentro de las restricciones se puede especificar cualquier declaración válida del lenguaje de predicados, que mezcle nombres de variables con los siguientes operadores:

1.– *IGUALDAD* (=). El valor de VARIABLE-1 = VARIABLE-2 es verdadero cuando ambas variables coinciden y es falso cuando son distintas.

3 = 5	NEGATIVO
4 = 4	AFIRMATIVO
6 + 3 = 3 × 3	AFIRMATIVO
"ROJO" = "Verde"	NEGATIVO
"LLUEVE" = "LLUEVE"	AFIRMATIVO

Figura 149. Toda expresión en el lenguaje de cálculo de predicados que esté compuesta mediante el comando IGUALDAD puede tomar dos valores: AFIRMATIVO o NEGATIVO. En la figura se incluyen algunos ejemplos de dicho comando.

2.– *DESIGUALDAD* (≠). El valor de VARIABLE-1 ≠ VARIABLE-2 es verdadero cuando ambas variables son distintas y es falso

cuando son iguales.

$3 \neq 5$	AFIRMATIVO
$4 \neq 4$	NEGATIVO
$6 + 3 \neq 3 \times 3$	NEGATIVO
"ROJO" $\neq$ "VERDE"	AFIRMATIVO
"LLUEVE" $\neq$ "LLUEVE"	NEGATIVO

Figura 150. El comando DESIGUALDAD produce el resultado contrario al del comando IGUALDAD. Es decir, el resultado es AFIRMATIVO si los valores comparados son distintos y NEGATIVO si son iguales.

3.– *CONJUNCION* ($\wedge$). El valor de EXPRESION-1 $\wedge$ EXPRESION-2 es verdadero cuando ambas expresiones son verdaderas y falso en cualquier otro caso.

$(3 = 3) \wedge$ ("ROJO" $\neq$ "VERDE")	AFIRMATIVO
$(3 + 3 = 6) \wedge$ ("A" = "A") $\wedge (2 \neq 4)$	AFIRMATIVO
("VERDE" = "ROJO") $\wedge (3 + 2 = 2 \times 3)$	NEGATIVO
$(6 = 3 + 3) \wedge$ ("LLUEVE" = "NIEVA")	NEGATIVO

Figura 151. El comando CONJUNCION ($\wedge$) permite verificar expresiones compuestas de forma que el resultado sólo es AFIRMATIVO si todas las expresiones componentes son AFIRMATIVAS.

4.– *DISYUNCION* ($\vee$). El valor de EXPRESION-1 $\vee$ EXPRESION-2 es verdadero cuando al menos una de las dos expresiones es verdadera y falso sólo cuando ambas expresiones son falsas.

$(3 = 3) \vee (\text{"ROJO"} = \text{"VERDE"})$ AFIRMATIVO

$(3 \neq 3) \vee (\text{"LLUEVE"} = \text{"NIEVA"})$ NEGATIVO

$(2 \times 3 = 6) \vee (24 = 92)$ AFIRMATIVO

Figura 152. Al ejecutar el comando DISYUNCION ($\vee$), el resultado de la expresión es AFIRMATIVO siempre que sea AFIRMATIVO el valor de al menos una de las expresiones elementales.

5.– *CUANTIFICADOR EXISTENCIAL* ($\exists$). El valor de $\exists$ × [EXPRESION] es verdadero cuando en la relación X, al menos existe un elemento para el que EXPRESION toma el valor verdadero.

RELACION-A

X	Y	Z
1	3	"A"
2	4	"B"
1	4	"A"
3	4	"B"

RELACION-B

V	W
"ROJO"	3
"VERDE"	9
"AZUL"	8

$\exists$ RELACION-A $[(X = 2) \wedge (Z = \text{"B"})]$ AFIRMATIVO
$\exists$ RELACION-B $(V = \text{"NEGRO"})$ NEGATIVO

Figura 153. El comando CUANTIFICADOR EXISTENCIAL ($\exists$) vale AFIRMATIVO cuando en la relación en que se ejecuta, al menos existe un elemento que verifique la expresión exigida.

6.– *CUANTIFICADOR UNIVERSAL* ($\forall$). El valor de $\forall$ X [EXPRESION] es verdadero cuando en la relación X todos los elementos verifican que EXPRESION.

RELACION-A

X	Y	Z
1	3	"A"
2	4	"B"
1	4	"A"
3	4	"B"

RELACION-B

V	W
"ROJO"	3
"VERDE"	9
"AZUL"	8

$\forall$ RELACION-A [(X = 1) $\vee$ (Z = "B")] AFIRMATIVO
$\forall$ RELACION-B (V = "AZUL") NEGATIVO

Figura 154. El comando CUANTIFICADOR UNIVERSAL ($\forall$) vale AFIRMATIVO sólo cuando todos los elementos de la relación en que se ejecuta, verifican la expresión impuesta.

A continuación veremos la utilización de los comandos más importantes del lenguaje ALFA, mediante ejemplos sobre la siguiente base de datos relacional:

Supongamos una base de datos relacional con datos referentes a suministradores, productos y cantidad suministrada con la siguiente estructura:

– RELACION DE SUMINISTRADORES:
S
S#, NOMBRE, CIUDAD, PROVINCIA

– RELACION DE PRODUCTOS:
P
P#, NOMBRE, COLOR, PESO

– RELACION DE CANTIDADES:
SP
S#, P#, CANTIDAD

Veamos algunas formas de obtener información de esta base de datos mediante el lenguaje ALFA.

1.– Para obtener una relación con todos los suministradores que viven en París, se solicitaría:

GET W (S.NOMBRE): S.CIUDAD = "PARIS".

Donde W es una zona de memoria destinada a almacenar el resultado de la consulta. Normalmente esta zona suele ser un array. Por ejemplo, si se utiliza lenguaje de programación PASCAL, la definición de W podría ser:

```
VAR: W: ARRAY [1..N] OF RECORD
                    NOMBRE: ARRAY [1..20]
                                OF CHAR;
                    END
```

2.– Si se desea obtener una representación completa de la relación de suministradores, se indicaría:

GET W (S)

3.– Si en cambio se desea obtener sólo parte de la información sin ninguna restricción, como una lista con los identificadores de los suministradores y su nombre, se solicitaría:

GET W (S.S#, S. NOMBRE)

4.– El número de restricciones que deben verificarse para la inclusión en el resultado es variable. Si se desea obtener una relación de todos los productos de color verde con un peso superior a 20, se solicitaría.

GET W (P. NOMBRE): (P. COLOR = "VERDE" ∧ P. PESO > 20)

5.– Las relaciones obtenidas en una consulta pueden estar clasificadas según un determinado atributo, bien ascendentemente (UP) o bien descendentemente (DOWN).

Por ejemplo, para obtener una relación de suministradores clasificada alfabéticamente por ciudades de la provincia de Madrid, se indicaría:

GET W (S.NOMBRE, S. CIUDAD, S. PROVINCIA): (S.PROVINCIA = "MADRID" ∧ UP S. CIUDAD).

6.– En determinados casos se puede desear obtener un único elemento de una relación. Por ejemplo, un producto de color rojo se solicitaría:

GET W (1) (P.P#, P.NOMBRE): P. COLOR = "ROJO".

7.– También se puede mezclar la obtención de un único elemento con la clasificación de la relación resultando, por ejemplo, para obtener un producto de color rojo y de peso máximo se indicaría:

GET W(1) (P.P#, P.NOMBRE) : (P. COLOR = "ROJO" ∧ DOWN P. PESO)

8.– La utilización de cuantificadores implica la definición de nombres externos de relaciones. Para ello se dispone de la instrucción RANGE que produce un nuevo nombre para una relación.

Por ejemplo si se desea obtener una lista con todos los suministradores que suministran la pieza 2 se solicitaría:

RANGE SP X
GER W(S.NOMBRE): $\exists \times$ [X. S # = S. S. # $\wedge$ X.P # = 2]

9.– Se pueden hacer preguntas con más de un cuantificador existencial, por ejemplo para obtener los nombres de los suministradores que suministran al menos una pieza roja, haríamos:

RANGE P X
RANGE SP Y
GET W (S. NOMBRE): $\exists \times$ [X.COLOR = "ROJO" $\wedge$ $\exists$ Y (Y.P # = X.P # $\wedge$ Y.S # = S.S #)]

que en forma normalizada sería:

RANGE P X
RANGE SP Y
GET W(S.NOMBRE): $\exists \times \exists$ Y [X.COLOR = "ROJO" $\wedge$ Y.P # = X.P # $\wedge$ Y.S # = S.S #]

10.– Las relaciones obtenidas como resultado de una consulta pueden contener información de varias relaciones originales, por ejemplo, para obtener para cada producto su número y las ciudades de quien las suministra, solicitaríamos:

RANGE SP X
GET W (P.P #, S.CIUDAD) : $\exists \times$ [X.S. # = S.S # $\wedge$ X.P. # = P.P #]

11.– También se pueden utilizar cuantificadores universales, por ejemplo, si queremos producir una lista con los nombres de los suministradores que no suministran el producto de clave 1:

RANGE SP X
GET W (S:NOMBRE) : $\forall \times$ [X.S # $\neq$ S.S # $\vee$ X.P # $\neq$ 1]

12.– Otro operador del cálculo de predicados que puede ser utilizado en las restricciones, es la implicación. Por ejemplo la misma pregunta del caso anterior se podría haber formulado de la siguiente manera:

RANGE SP X
GET W (S.NOMBRE) : $\forall \times$ [X.S # = S.S # $\Longrightarrow$ X.P # $\neq$ 1]

13.– También existen funciones de cálculo, como la instrucción COUNT que cuenta el número de elementos de un conjunto o la función TOTAL que calcula la suma de elementos numéricos de un conjunto.

El número de funciones definidas es muy amplio, no obstante si se desea se puede implementar cualquier nueva función.

14.– Hasta ahora todos los comandos descritos se utilizaban para la obtención de información. También existen tres operaciones básicas de almacenamiento:

Para incluir una n-tupla se utiliza el comando PUT; para borrarla DELETE, y para actualizarla UPDATE.

Antes de utilizar cualquiera de estas tres instrucciones, es necesario buscar los valores a actualizar, y para ello se necesita el comando HOLD.

• Por ejemplo, para cambiar el color del producto 1 haríamos:

```
HOLD W (P.P.#, P.COLOR): P.P# = 1
W. COLOR = "AMARILLO"
UPDATE
```

Un problema que surge en las actualizaciones, es que sólo es posible modificar una única relación simultáneamente, con lo que se genera un problema temporal de inconsistencia si hay que actualizar un mismo dato en varias relaciones, al tener que hacerlo de una en una.

• Para añadir un nuevo producto se haría:

```
W. P # = 7
W.NOMBRE = "GRUA"
W.COLOR = "AZUL"
W.PESO = 15
PUT W (P)
```

Mediante esta operación se comprueba que la clave de la nueva n-tupla, no existe previamente en la relación de productos.

• Para eliminar el suministrador 1 se haría:

```
HOLD W (S): S.S# = 1
DELETE W
```

Si el campo que se desea cambiar es el identificador de la n-tupla, es necesario hacerlo en dos pasos distintos: primero se elimina la n-tupla y a continuación se realiza la modificación.

• Por ejemplo para cambiar el identificador del producto 3 por 7, se haría:

```
HOLD W(P) : P.P # = 3
DELETE W
W.P # = 7
PUT W(P)
```

6.– LENGUAJE QBE (QUERY-BY-EXAMPLE)

Este lenguaje de consulta para bases de datos relacionales se utiliza generalmente mediante una pantalla. El usuario selecciona por medio de un "menú", los nombres de las relaciones que quieren utilizar y en la pantalla aparecen las cabeceras (nombres de los dominios) de las relaciones indicadas sobre columnas varias, que el usuario puede rellenar con un ejemplo del resultado deseado. Si se quieren seleccionar las tuplas en las que un cierto dominio toma un determinado valor, se escribe dicho valor en la columna correspondiente, mientras que si escribe algo subrrayado hace las veces de variable o ejemplo. Un delimitador (P.) indica los campos que se desean recuperar.

Supongamos la misma base relacional sobre la que hicimos los ejemplos del lenguaje ALFA para realizar algunos ejercícios con el lenguaje QBE:

– RELACION DE SUMINISTRADORES
S
S#, NOMBRE, CIUDAD, PROVINCIA

– RELACION DE PRODUCTOS
P
P#, NOMBRE, COLOR, PESO

– RELACION DE CANTIDADES
SP
S#, P#, CANTIDAD

Veamos algunas formas de obtener información de esta base de datos mediante el lenguaje QBE.

1.– Para obtener una lista de todos los números de productos:

P	P #	NOMBRE	COLOR	PESO
	P. X			

2.– Para obtener toda la información referente a todos los suministradores:

S	S #	NOMBRE	CIUDAD	PROVINCIA
	P.3	P. LUIS	P. MADRID	P. HUESCA

3.– Para obtener las claves de suministradores de PARIS:

S	S #	NOMBRE	CIUDAD	PROVINCIA
	P.4	P. ANTONIO	PARIS	

4.– Para obtener los nombres de los suministradores de PARIS o cuya clave sea mayor que 20:

S	S #	NOMBRE	CIUDAD	PROVINCIA
		P. ANTONIO	PARIS	
	> 20	P. JUAN		

5.– Para obtener una lista de las claves de los suministradores que suministran el producto 1 o el producto 2:

SP	S #	P #	CANTIDAD
	P.3	1	
	P.4	2	

6.– Para obtener una lista de las claves de los suministradores que suministran el producto 1 y 2:

SP	S #	P #	CANTIDAD
	P.3	1	
	P.3	2	

7.– Si lo que desea es una lista de los nombres de los suministradores que suministran el producto 3, harían falta dos relaciones:

S	S #	NOMBRE	CIUDAD	PROVINCIA
	4	P. JUAN		

SP	S #	P #	CANTIDAD
	4	3	

8.– Para obtener una lista con los nombres de los suministradores que suministran al menos un producto de color azul, sería necesario utilizar tres relaciones:

P	P #	NOMBRE	COLOR	PESO
	3		AZUL	

S	S #	NOMBRE	CIUDAD	PROVINCIA
	5	P. PEDRO		

SP	S #	P #	CANTIDAD
	5	3	

9.– Si se desea obtener un listado de los nombres de los suministradores que no suministran el producto 5.

S	S #	NOMBRE	CIUDAD	PROVINCIA
	3	P. JULIAN		

SP	S #	P #	CANTIDAD
7	3	5	

10.– El sistema para realizar modificaciones sobre la base de datos "funciona" igual que en el lenguaje ALFA, con las palabras clave INSERT (en vez de PUT), DELETE y UPDATE se produce la inserción, el borrado y la modificación respectivamente.

Por ejemplo, para cambiar el color rojo por el amarillo en todos los productos:

	P	P #	NOMBRE	COLOR	PESO
		3		ROJO	
UPDATE		3		AMARILLO	

Si queremos sumar 100 a la cantidad suministrada por el suministrador 7 del producto 3:

	SP	S #	P #	CANTIDAD
		7	3	X
UPDATE		7	3	X + 100

Para insertar un nuevo producto:

P	P #	NOMBRE	COLOR	PESO
INSERT	12	TORNILLO	GRIS	10

Y para suprimir al suministrador de clave 7.

S	S #	NOMBRE	CIUDAD	PROVINCIA
DELETE	7			

11.– En el lenguaje QBE también existen funciones predefinidas, por ejemplo para contabilizar el número de suministradores que viven en Toledo haríamos lo siguiente:

S	S #	NOMBRE	CIUDAD	PROVINCIA
COUNT			TOLEDO	

y para contabilizar el número total de suministradores:

S	S #	NOMBRE	CIUDAD	PROVINCIA
COUNT			TOLEDO	

CUESTIONARIO CAPITULO 10

UTILIZACION DE BASES DE DATOS RELACIONALES

1.– ¿Cómo se define una relación?
2.– ¿Qué es una N-Tupla?
3.– ¿A qué se denomina atributo?
4.– ¿Qué diferencia hay entre esquema y subesquema?
5.– ¿Qué es una DF?
6.– ¿Cuándo una cobertura no es redundante?
7.– ¿Qué es una superclave?
8.– ¿Qué se entiende por normalización?
9.– ¿Qué anomalías pueden presentarse en una base de datos relacional?
10.– ¿Qué es una Primera Forma Normal?
12.– ¿Y una Segunda Forma Normal?
13.– ¿Y la Tercera Forma Normal?
14.– ¿Y la Cuarta?
15.– ¿Y la Quinta?
16.– ¿Qué es el lenguaje ALFA?
17.– ¿Qué es un lenguaje Query?

APENDICE

PRINCIPALES BASES DE DATOS DEL MERCADO

En este último capítulo del libro, vamos a intentar recopilar una información somera sobre los principales productos que se pueden encontrar, para la gestión de bases de datos.

La tarea de realizar una guía completa de un producto tan diversificado como el de las bases de datos (raro es el fabricante que no dispone de uno o más sistemas para cada uno de sus modelos), sería impropio de un texto, que tan solo pretende realizar una introducción al concepto general "base de datos"; por lo tanto, en el caso de que el lector pretenda tomar una decisión sobre qué base de datos es más adecuada para sus necesidades, le recomendamos que acuda a otras fuentes (dentro de esta misma colección) que le orienten más adecuadamente.

El proceso seguido para la elaboración de este capítulo, ha comenzado con una relación de quince productos distintos, haciendo especial hincapié en aquellos que pueden ser utilizados en equipos medios o grandes, aunque no olvidando los que han sido preparados para equipos pequeños. A continuación se solicitó información a los propios fabricantes, por lo tanto, los datos suministrados para cada producto son de tipo comercial y no técnico, y por último, se elaboraron unas "fichas" para cada uno de ellos en las que se indica: el nombre comercial del producto, el tipo de ordenadores en que puede ser utilizado, el nombre del fabricante y/o distribuidor, el entorno de funcionamiento, y unos breves comentarios.

A continuación, para no utilizar ningún criterio subjetivo en la clasificación, se incluyen las citadas "fichas" en orden alfabético.

SOFTWARE DE BASES DE DATOS

- **Nombre comercial:** ADABAS
- **Tipo de equipos en que es utilizable:** Medios y grandes.
- **Fabricante:** FRASER ESPAÑOLA (SOFTWARE AG)
- **Entorno de funcionamiento:** IBM y compatibles, SIEMENS 4004/45 y superiores, y DEC PDP 11.
- **Comentarios:** Para el desarrollo de aplicaciones con esta base de datos, puede utilizarse el lenguaje NATURAL y un monitor de transacciones denominado COMPLETE.
  Cuando se utiliza en entorno IBM puede ser compatible con diversos productos como: CICS/VS, o IMS/DC.

- **Nombre comercial:** CA-DBMS
- **Tipo de equipos en que es utilizable:** Pequeños.
- **Fabricante:** COMPUTER ASSOCIATES.
- **Entorno de funcionamiento:** Sistemas operativos: CP/M y MS-DOS.
- **Comentarios:** Este sistema está diseñado inicialmente para el equipo CA-EXECUTIVE.
  Puede simultanearse con productos de ayuda, para la generación de formatos en pantalla como el CA-FORM y para el tratamiento de gráficos como el CA-GRAF.
  Su funcionamiento se basa en el modelo relacional.

- **Nombre comercial:** CA-UNIVERSE
- **Tipo de equipos en que es utilizable:** Medios y grandes.
- **Fabricante:** COMPUTER ASSOCIATES.
- **Entorno de funcionamiento:** IBM y compatibles.
- **Comentarios:** Esta base de datos es una generalización de la anterior, para su utilización en equipos IBM.
  Es compatible con otros productos desarrollados por el mismo fabricante como el CA-EXECUTIVE y CA-DBMS.
  En la actualidad este paquete está anunciado pero no disponible.

- **Nombre comercial:** DATA COM/DB
- **Tipo de equipos en que es utilizable:** Medios y grandes.
- **Fabricante:** APPLIED DATA RESEARCH.
- **Entorno de funcionamiento:** IBM y compatibles.
- **Comentarios:** El fabricante original de este producto fué INSYTE DATACOM CORPORATION que, posteriormente fué comprada por APPLIED DATA RESEARCH.
  Es compatible con el monitor de teleproceso DATA COM/DC y dispone de un generador de informes denominado DATA REPORTER.

- **Nombre comercial:** DBASE (I, II y III)
- **Tipo de equipos en que es utilizable:** Pequeños.
- **Fabricante:** ASHTON TATE.
- **Entorno de funcionamiento:** Sistemas operativos CP/M y MS-DOS.
- **Comentarios:** Puede utilizarse simultáneamente con diversos productos, como:
  El generador de programas QUICK CODE.
  El sistema de gestión de ficheros FRIDAY.
  El paquete de utilidades DUTIL.
  El software de tratamiento de gráficos DGRPH.

- **Nombre comercial:** DL/I
- **Tipo de equipos en que es utilizable:** Medios y grandes.
- **Fabricante:** IBM.
- **Entorno de funcionamiento:** Sistemas operativos IBM, DOS, OS, MVS y compatibles.
- **Comentatios:** Es probablemente uno de los dos productos más utilizados para la gestión de bases de datos. Funciona indistintamente con CICS/VS o IMS/DS.
  La relación de productos compatibles sería practicamente interminable.
  En capítulos anteriores de este mismo libro se puede estudiar con más detalle.

- **Nombre comercial:** IDMS
- **Tipo de equipos en que es utilizable:** Medios y Grandes.
- **Fabricante:** TEG IBERICA (CULINAME).
- **Entorno de funcionamiento:** IBM y compatibles.
- **Comentarios:** Parece un diccionario de datos IDD muy recomendable, tanto para el desarrollo de aplicaciones como para la documentación.
  También dispone de un sistema de preguntas interactivas y un generador automático de programas denominado INTERACT.

- **Nombre comercial:** IDS II
- **Tipo de equipos en que es utilizable:** Medios y Grandes.
- **Fabricante:** BULL.
- **Entorno de funcionamiento:** BULL.
- **Comentarios:** Puede ser utilizado junto con un monitor de teleproceso llamado TDS.
  Para la consulta directa sobre la información almacenado en la base de datos se puede utilizar un lenguaje de consulta QUERY.

- **Nombre comercial:** IMAGE
- **Tipo de equipos en que es utilizable:** Medios y Grandes.
- **Fabricante:** HEWLET PACKARD.
- **Entorno de funcionamiento:** HEWLET PACKARD MP-250 y MP-3000.
- **Comentarios:** Este sistema diseñado por HEWLET para sus equipos medios y grandes, es compatible con la mayoría de productos del propio fabricante, tales como FORMS para la definición de formatos en pantalla, programas gráficos, etc.

- **Nombre comercial:** INFO
- **Tipo de equipos en que es utilizable:** Medios y Grandes.
- **Fabricante:** PRIME.
- **Entorno de funcionamiento:** PRIME.
- **Comentarios:** Esta base de datos basada en el modelo relacional, estaba diseñada inicialmente para la gama pequeña de IRIME. Más adelante se amplió el producto hasta llegar a ser compatible con los principales paquetes de PRIME, como MEDUSA para el tratamiento de gráficos, FORMS para la definición de formatos, etc.

- **Nombre comercial:** MAPPEX 80 y 1100
- **Tipo de equipos en que es utilizable:** Medios y Grandes.
- **Fabricante:** SPERRY.
- **Entorno de funcionamiento:** SPERRY 5/80 y 1100.
- **Comentarios:** El software de base denominado MAPPER, tiene su principal utilidad, como ayuda para el desarrollo interactivo de aplicaciones con estructura de base de datos.

- **Nombre comercial:** RAMIS (I y II)
- **Tipo de equipos en que es utilizable:** Medios y Grandes.
- **Fabricante:** MATHEMATICA PRODUCTS GROUP.
- **Entorno de funcionamiento:** IBM y compatibles, BURROUGHS.
- **Comentarios:** Este producto resulta de la conjunción de un lenguaje no procedural con un sistema de gestión de bases de datos propiamente dicho.

- **Nombre comercial:** SQL/DS
- **Tipo de equipos en que es utilizable:** Medios y Grandes.
- **Fabricante:** IBM.
- **Entorno de funcionamiento:** IBM DOS/VSE y compatibles.
- **Comentarios:** Esta base de datos modelo relacional, puede utilizarse para el desarrollo de aplicaciones o en un sentido más general. Para la gestión interactiva puede utilizarse simultáneamente con el paquete ISPF.

- **Nombre comercial:** SUPERFILE.
- **Tipo de equipos en que es utilizable:** Medios y Grandes.
- **Fabricante:** SOUTHDATA.
- **Entorno de funcionamiento:** Sistemas operativos CP/M, MS-DOS, TURBO-DOS y UNIX.
- **Comentarios:** Los principales productos compatibles con el SUPERFILE son el sistema SUPERFORMS para la generación de formatos en pantalla y el generador de informes SUPERTAB.

- **Nombre comercial:** TOTAL
- **Tipo de equipos en que es utilizable:** Medios y Grandes.
- **Fabricante:** ERIA (CINCOM SYSTEM).
- **Entorno de funcionamiento:** IBM, DIGITAL, NCR, SPERRY, BULL, SIEMENS, ICL, CONTROL DATA CORPORATION, PRIME, BURROUGHS y PERKIN ELMER.
- **Comentarios:** Junto con el DL/I, este es el sistema más utilizado para la gestión de bases de datos. Por lo tanto es compatible con numerosos productos como los monitores de teleproceso CICS/VS, ENVIRON, etc. Dispone de un programa para la recuperación de la información denominado SOCRATES y un lenguaje de consulta interactiva llamado T-TASK.

BIBLIOGRAFIA

1.– R. W. Engles, "A Tutorial on Data Base Organization". IBM Corporation Technical Report TR.00.2004. IBM, Paughkeeprie, N.Y. 1.970

2.– R. Canning, "The Debate on Data Base Management" EDP Analyzer, 1.972.

3.– C.J. Date y P. Hapewell, "File Definition and Logical Data Independence", Proc. ACM SIGFIDET Workshop on Data Description, Access and Control, 1.971.

4.– C.J. Date y P. Hopewell, "Storage Structure" and Physical Data Independence" ACM SIGFIDET Workshop on Data Description, Access and Control, 1.971.

5.– CODASYL Systems Committee, "Feature Analysis of Generalized Data Base Management Systems" ACM, 1.971.

6.– A. M. MEALY, "Another Look at Data", Proveedings of the AFIPS 1.967 Fall Joint Computer Conference, Val 31, AFIPS Press 1.968.

7.– JAMES MARTIN, "Security, Accuracy and Privacy in Computers Systems" Prentice-Hall Inc. 1.974.

8.– KNUTH, Donald E., "The Art of Computer Programming" Volume 1, "Fundamental Algorithms". Addison-Wesby, 1.968.

9.– C.J. DATE, "Principles of Data Base Systems". Addison-Wesby.

10.– IBM, "Data Language/I System/370 DOS/VS, General Information Manual", 1.974.

11.– IBM, "Information Management Systems Virtual Storage (IMS/

VS), General Information Manual", 1.974.

12.– E. F. CODD, "A Relational Model of Data for Large Shared Data Banks", ACM, 1.970.

13.– E. F. CODD, "Further Normalization of the Data Base Relational Model", Prentice-Hall, 1.972.

14.– P. A. BERNSTEIN, "Synthesizing Third Normal Form Relations from Functional Dependencies", ACM, 1.976.

15.– A. J. SYMONDS, R. A. LORIE, "A Schema for Describing a Relational Data Base" ACM-SIGFIDET, 1.970.

16.– M. M. ZLOOF, "Query by Example", Proc. National Computer Conf. 1.975.

INDICE

INDICE

Presentación 5
Autor 7

PRIMERA PARTE
FICHEROS

Capítulo 1. INTRODUCCION A LOS FICHEROS
1.– Antecedentes 11
2.– Elementos de información 12
3.– Registros y ficheros 13
4.– Tipos de ficheros 19
5.– Diseño de ficheros 22
6.– Características de los ficheros 24
Cuestionario 26

Capítulo 2. SOPORTES DE FICHEROS
1.– Introducción 27
2.– Medios de acceso secuencial 28
3.– Medios de acceso directo 37
4.– Uso de los soportes 43
Cuestionario 44

Capítulo 3. ORGANIZACION DE FICHEROS
1.– Organización lógica y organización física 45
2.– Organización secuencial 48
3.– Organización de archivos standard 49
4.– Organización directa o de archivos relativos 50
5.– Organización encadenada 51
6.– Organizaciones especiales derivadas del encadenado .. 54

7.– Organización regionalizada 58
8.– Organización particionada 58
9.– Organización secuencial-indexada 58
10.– Organización con fichero inverso 60
11.– Conclusiones 62
Cuestionario 64

Capítulo 4. METODOS DE ACCESO A FICHEROS
1.– Introducción 65
2.– Método de acceso secuencial (SAM) 66
3.– Acceso directo (DAM) 71
4.– Método de acceso secuencial-indexado (ISAM) 78
5.– Método de acceso particionado (PAM) 82
6.– Método de acceso de memoria virtual (VSAM) 84
7.– Elección del método de acceso 89
Cuestionario 92

SEGUNDA PARTE
BASES DE DATOS

Capítulo 5. CONCEPTOS FUNDAMENTALES SOBRE BASES DE DATOS
1.– Orígenes históricos 95
2.– Sistemas de bases de datos 98
3.– Utilización de bases de datos 101
4.– Redundancia 104
5.– Modificaciones en la organización 106
6.– Necesidades de seguridad 108
7.– Independencia de datos 109
8.– Principales ventajas de la utilización de bases de datos .. 111
Cuestionario 116

Capítulo 6. ORGANIZACION DE LA INFORMACION
1.– Información 117
2.– Información y ficheros tradicionales 119
3.– Información y bases de datos 122
4.– Relaciones entre los datos 125
5.– Visiones de la base de datos 130
6.– Correspondencia entre datos 132
Cuestionario 136

Capítulo 7. MODELOS DE BASES DE DATOS
1.– Modelo jerárquico 137
2.– Modelo de red 145
3.– Modelo relacional 152
Cuestionario 163

Capítulo 8. UTILIZACION DE BASES DE DATOS JERARQUICAS
1.– Principales lenguajes de datos 165
2.– Bases de datos IMS 169
3.– Otras bases de datos jerárquicas (Modelo Codasyl) ... 181
Cuestionario 189
Capítulo 9. UTILIZACION DE BASES DE DATOS EN RED
1.– Similitud entre los modelos jerárquico y de red 191
2.– Lenguajes para bases de datos en red 193
Cuestionario 200
Capítulo 10. UTILIZACION DE BASES DE DATOS RELACIONALES
1.– Conceptos generales 201
2.– Definiciones 202
3.– Dependencia funcional entre atributos 207
4.– Normalización 209
5.– Lenguaje ALFA 218
6.– Lenguaje QBE (Query By Example) 225
Cuestionario 230
Apéndice. PRINCIPALES BASES DE DATOS DEL MERCADO .. 231
Bibliografía 239
Indice 241

TITULOS PUBLICADOS:

- ☐ Introducción a la Informática.
- ☐ Cómo seleccionar su ordenador.
- ☐ Cómo adquirir su programa informático (Selección de programas).
- ☐ Teleinformática.
- ☐ Cómo informatizar su empresa.
- ☐ Diccionario de Informática.

2968
28/00

1
3
7
24
29
48
2

ARCHIVOS, FICHEROS Y BASES DE DATOS INFORMATICOS